THESIS

FINITE ELEMENT BASED STRUCTURAL DAMAGE DETECTION USING ARTIFICIAL BOUNDARY CONDITIONS

by

Rafael A. Lagunes Arteaga

September 2007

Thesis Advisor: Joshua H. Gordis

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Finite Element Based Structural Damage Detection Using Artificial Boundary Conditions.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Rafael A. Lagunes Arteaga			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Finite element models can be used to discern the location and severity of damage in structures. This is frequently pursued by using the differences in measured and computed natural frequencies, in conjunction with the sensitivities calculated using the FE model. Given that a modal test produced a limited number of natural frequencies for a structure, the concept of Artificial Boundary Conditions (ABC) was developed, which yields additional natural frequency information for a structure. This is accomplished by artificially imposing additional boundary conditions to the measured data. In this thesis, the use of ABC to produce an improved set of structural sensitivities is explored. It is shown that the selection of ABC sets is best guided by the strain energy distribution in the structure.

14. SUBJECT TERMS Artificial Boundary Conditions, Finite Element Model, Natural Frequencies, Sensitivity Based Updating, Strain and Kinetic Energies.			**15. NUMBER OF PAGES** 183
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

FINITE ELEMENT BASED STRUCTURAL DAMAGE DETECTION USING ARTIFICIAL BOUNDARY CONDITIONS

Rafael A. Lagunes Arteaga
Lieutenant Mexican Navy
B.S., Naval Engineering, Naval Academy, Mexico, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2007

Author:	Rafael A. Lagunes Arteaga
Approved by:	Joshua H. Gordis Thesis Advisor
	Anthony J. Healey Chairman, Department of Mechanical and Astronautical Engineering

ABSTRACT

Finite element models can be used to discern the location and severity of damage in structures. This is frequently pursued by using the differences in measured and computed natural frequencies, in conjunction with the sensitivities calculated using the FE model. Given that a modal test produced a limited number of natural frequencies for a structure, the concept of Artificial Boundary Conditions (ABC) was developed, which yields additional natural frequency information for a structure. This is accomplished by artificially imposing additional boundary conditions to the measured data. In this thesis, the use of ABC to produce an improved set of structural sensitivities is explored. It is shown that the selection of ABC sets is best guided by the strain energy distribution in the structure.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my sincerest appreciation to Professor Joshua H. Gordis for his unparalleled understanding, and patience as well as dedicated support throughout this task.

I would also like to thank Zoe, my wife, for her overwhelming love and support during this challenging educational endeavor.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Finite element models are used to predict the dynamic response of the system to various excitations, boundary conditions and parameter changes. When tests are performed to validate the FE model, inevitably their results, commonly natural frequencies and mode shapes, do not coincide with the expected results from the FE model. These discrepancies stem from uncertainties regarding the governing equation of the system, simplifying assumptions in the derivation, and inaccurate boundary conditions. Clearly one would like to have a better model, based on both the theoretical and the experimental results. The problem of how to modify the theoretic model from the experimental results is known as model updating (Kenigsbuch and Halevi, 1998).

Finite Element Models are defined by a large number of physical parameters, but a modal test of a structure results in a small number of modal parameters which are used to modify the model parameters. That is why updating a finite element model in order to produce a reliable, confident prediction of the structural response of a system tends to become a difficult task to be accomplished.

A traditional problem associated with updating a FEM has been obtaining sufficient measurements in order to obtain accurate enough results. In theory it should be possible to measure as many mode shapes as required. This is not feasible, because the available transducers and data acquisition hardware limit the frequency range that can be measured. Also, as the frequency increases so does the modal density, causing considerable difficulty for the modal extraction algorithms. Even if these difficulties could be overcome it would still be impossible to accurately measure the infinite number of modes of a structure. A finite element model only accurately predicts the lower third or so of the natural frequencies and mode shapes. Therefore the structure could not be expected to reproduce all the modes predicted by a model (Ewins, 2000).

FEM can be very large, extending in many cases to several hundred thousand Degrees of Freedom (DOF), or more. Experimental modal analysis rarely uses more than a couple of hundred transducers; consequently not all the degrees of freedom in the analytical model will be measured. There are often internal nodes which cannot be measured. There are two ways of overcoming this difficulty, namely reducing the analytical model order or expanding the measured mode shapes.

Procedures have been developed for measuring larger experimental databases. One method is to identify additional and distinct mode frequencies from the same modal test that was already performed to identify the mode frequencies of the standard system, without the need for physical modification of the structure. The additional frequencies and mode shapes correspond exactly to the mode frequencies found when combinations of measured coordinates are constrained to ground. These additional frequencies are therefore associated with different boundary conditions for the structure and no physical change in the boundary conditions has been made. (Gordis, 1999).

Since the boundary conditions are not actually physically applied, they are termed Artificial Boundary Conditions (ABC). The ABC are imposed on the FEM and correspond to ideal constraints. With only one experimental database, this design yields a separate FEM for each ABC configuration. With the exception of the imposed boundary conditions, each model is identical.

In a set of spatially incomplete Frequency Response Function (FRF) data, the ABC are the constraints that define an Omitted Coordinate System (O-SET). In performing a vibration test, a choice is made as to the set of coordinates to instrument with response transducers. This set of DOF correspond to the coordinates that are going to be actually measured.

Sensitivity based updating makes use of the modal parameters to help locate a difference between a FEM and an existing structure. The parameters experimentally determined from the actual structure are the natural frequencies of the structure, the corresponding mode shapes, and the modal damping ratios; the parameters needed from the initial FE model are the natural frequencies and the associated frequency sensitivities. Finite element model sensitivities are the first derivative of the eigen-problem solution set with respect to the design variables under consideration.

Frequency sensitivities are the basis for a linear approximation to compute the change in the natural frequencies of a model based on a change in a given design variable. Trying to locate the source of this difference, i.e. either the error in the model or damage in the structure, is a problem that is generally underdetermined due to the number of parameters that can be altered and the limited number of measured parameters. With the application of ABC's a better solution to this problem can often be formulated. This method of frequency updating greatly reduces the computational time required to calculate the results of a change to large finite element models by eliminating the need to resolve the eigen-problem for each iteration. This process has been used extensively for model updating (see for example, Flanigan, 1998, Luber, 1995, Dascotte, 1995).

Furthermore, there are some other approaches to search within the solution domain for potential solutions, such as the Constrained/ Unconstrained Optimization Routines. The Optimization Routine employed searches for the minimum value of an objective function which is constructed to minimize the differences between the analytical model parameters and the experimental data. This organized approach of searching for the optimal combination of design variables, minimizes the number of combinations that are investigated thereby reducing the computational requirements of this process.

The real difficulty to this method of correcting a finite element model is to determine which candidate solution most closely matches the experimental data. Multiple combinations of design variable changes can result in the same changes to the natural frequencies of the model so the accuracy of the frequencies is not the only evaluation factor.

An improved method of solution evaluation is to determine the effects on the analytical mode shapes and compare them to the experimental mode shapes. This comparison of mode shapes is known as the Modal Assurance Criterion (MAC) and is discussed in (Allemang, 1982). The MAC is a measure of how closely the analytical and the experimental mode shapes coincide. The MAC has a value from 0 to 1, where the candidate solution with a MAC value closer to one when compared to the experimental data most closely resembles the experimental case and is selected as the optimal solution.

The more meticulous the Objective Function is assembled, that is, the more properties of the structure are considered (design variables), the better the error predictions are to be found in the Constrained Optimization Routine.

Three different Objective Functions were evaluated in the present thesis in order to correlate the confidence in the damage detection results.

II. THEORY

The matrix equation of motion for a spring mass oscillator (SMO) will represent the starting point of the derivation:

$$[M]\{\ddot{x}\}+[C]\{\dot{x}\}+[K]\{x\}=\{F(t)\} \tag{2.1}$$

Excluding the damping $[C]$, equation (2.1) becomes:

$$[M]\{\ddot{x}\}+[K]\{x\}=\{F(t)\} \tag{2.2}$$

Rearranging equation (2.2) in a more explicit form, it is written as:

$$\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix}\begin{Bmatrix} \ddot{x}_1(t) \\ \\ \ddot{x}_n(t) \end{Bmatrix}+\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix}\begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix}=\begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.3}$$

where [M] and [K] represent the mass and stiffness matrices which define the physical properties of the structure. Both matrices are square and symmetric, that is of size *n* x *n* where n is the number of DOF in the model. The vector $\{F(t)\}$ represents the excitation vector of size *n* x *1* along the x coordinate.

Any continuous system has a continuous arbitrary distribution of mass and stiffness; hence it also comprises an infinite number of DOF. During modal tests, only a finite number of DOF are measured, and these measured coordinates represent the analytical set or (ASET), where the measuring instruments (accelerometers) are positioned. The remaining unmeasured DOF are considered the omitted coordinate set or (OSET).

A. SPATIALLY COMPLETE AND INCOMPLETE DATA

It is not physically possible to develop a complete finite element model of a structural system over a frequency domain. This would require a number of measuring devices (accelerometers) to be equal to the number of DOF of the finite element model, hence generating it to have a *determined* problem where the numbers of equations is equal to the number of unknowns, therefore dealing with a set of spatially complete data to solve the problem with reliable accuracy. However, practical realities only allow a limited number of DOF to be measured in a modal test. Hence the result for a real world structure is a measured Frequency Response Function (FRF) matrix that is spatially incomplete, involving fewer measured DOF than model DOF, therefore obtaining an *underdetermined* problem.

For this reason, to perform the structural system analysis, that is either the model updating or the damage detection, the analytical mode shapes of the Frequency Response Function matrix must be reduced in size such that only the mode shapes identified from the test/ measured Frequency Response Function matrix are the ones employed to perform the system analysis; this is done via a Reduced Order Model.

The Reduced Order Model makes use of the ASET and OSET coordinates, where as stated before, the OSET coordinates are not associated with the measurement locations on the structure, and the reduced Frequency Response Function matrix is composed of response information from the limited number of measurements corresponding to the ASET coordinates. It is important to note that the reduced FRF matrix implicitly defines a dynamically reduced Impedance matrix, furthermore, because the basic system has not been altered and that it still has the same number of DOF, even though it was determined not to describe all of the DOF, the elements remaining in the reduced FRF matrix are identical to the corresponding elements of the full FRF matrix. (Ewins, 1984). The matrix still contains all the modal information of a fully described matrix.

B. ANALYTICAL AND OMITTED COORDINATE SETS

Rearranging equation (2.3) into the frequency domain for the steady-state harmonic response for a full order model without the damping, it becomes:

$$\left[\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix}\right] * \begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.4}$$

where Ω is the frequency of the harmonic excitation. Considering that the number of DOF's are comprised for the measured and unmeasured set of coordinates, that is the ASET and OSET coordinates, we can express equation (2.4) in the following form:

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_o \end{Bmatrix} \tag{2.5}$$

where the subscript "a" refers to the ASET coordinate and "o" to the OSET coordinate. Equation (2.5) is also known as the impedance model and it can be redefined as:

$$\begin{bmatrix} Z_{aa} & Z_{ao} \\ Z_{oa} & Z_{oo} \end{bmatrix} \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_o \end{Bmatrix} \tag{2.6}$$

where [Z] represents the system impedance matrix evaluated at the forcing frequency, Ω. Assuming a static constraint, that is no excitation on the omitted coordinates then $\{f_o\} = \{0\}$, and equation (2.5) is rewritten as:

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ 0 \end{Bmatrix} \tag{2.7}$$

Rearranging equation (2.7) into two separate equations and substituting the OSET coordinates in terms of the ASET coordinates, only one equation is required:

$$[K_{oa}]\{x_a\} + [K_{oo}]\{x_o\} - \Omega^2 [[M_{oa}]\{x_a\} + [M_{oo}]\{x_o\}] = 0 \tag{2.8}$$

Solving equation (2.8) in terms of $\{x_o\}$ it becomes:

$$\{x_o\} = \left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1} \left[- K_{oo}^{-1} K_{oa} + \Omega^2 K_{oo}^{-1} M_{oa}\right] \{x_a\} \tag{2.9}$$

or

$$\{x_o\} = \left[-Z_{oo}\right]^{-1} \left[Z_{oa}\right] \{x_a\} \tag{2.10}$$

By definition the inverse of the OSET impedance matrix $\left[Z_{oo}\right]$ can be expressed as:

$$\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1} = \frac{1}{Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]} Adj\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] \tag{2.11}$$

where the Det[$\bullet$] indicates the determinant and Adj[$\bullet$] indicates the adjoint matrix. From equation (2.11), it is noticeable that the bracketed inverse term does not exist at those frequencies which satisfy:

$$Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] = 0 \tag{2.12}$$

By definition the eigenvalues which satisfy equation (2.12) are those defined by the $\left[K_{oo}\right]$ and $\left[M_{oo}\right]$ matrices, hence the resulting frequencies correspond to the OSET frequencies only.

Considering that the eigenvalues and eigenvectors of a system are defined by the unrestrained DOF of the corresponding system, the OSET coordinates of the OSET system are unrestrained and the ASET coordinates are constrained to ground.

The FRF matrix derivation, as well as the Reduced Order Model are described in the following sections.

C. REDUCED ORDER MODEL

The process of instrumenting a structure with a finite number of response transducers defines a Reduced Order Model, where the impedance of the Reduced Order Model is nonlinearly dependent on the impedance of the Full Order Model (Gordis, 1996).

In order to perform model updating using spatially incomplete data, a reduction of the model data is done, that is the reduction of the analytical model, with no loss of generality of the results.

Considering a full order "exact" FRF model of a structure, we would have a FRF matrix of infinite dimensions:

$$[H] = \begin{bmatrix} H_{aa} & H_{ao} \\ H_{oa} & H_{oo} \end{bmatrix} \tag{2.13}$$

where the experimental FRF matrix actually measured in a real test is seen to be a matrix partition which has been extracted from the infinite dimension matrix, i.e.

$$\left[\overline{H^{x}}\right] = [H_{aa}] \tag{2.14}$$

The overbar in equation (2.14) specifies a reduced model, the superscript x denotes an experimentally measured FRF, and the $[H_{aa}]$ matrix represents a structural dynamic model that has been reduced using exact dynamic reduction (Gordis, 1996). From the partitioned inverse relation of the FRF matrix to its associate impedance, i.e. $[Z][H] = I$, we have:

$$\begin{bmatrix} Z_{aa} & Z_{oa} \\ Z_{ao} & Z_{oo} \end{bmatrix} \begin{bmatrix} H_{aa} & H_{oa} \\ H_{ao} & H_{oo} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.15}$$

Now, solving for $[H_{aa}]$ gives:

$$[H_{aa}] = \left[Z_{aa} - Z_{ao} Z_{oo}^{-1} Z_{oa}\right]^{-1} \tag{2.16}$$

Since it was assumed no excitation on the omitted coordinates ($\{f_o\}=\{0\}$), the exact dynamic relationship between the OSET coordinates and the ASET coordinates gives:

$$\begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{bmatrix} I \\ -Z_{oo}{}^{-1} Z_{oa} \end{bmatrix} \{x_a\} \tag{2.17}$$

Rearranging equation (2.5) using equation (2.17), the dynamic reduced model results:

$$\{f_a\} = \left[Z_{aa} - Z_{ao} Z_{oo}^{-1} Z_{oa} \right] \{x_a\} \tag{2.18}$$

Comparing equations (2.16) and (2.18), we can see that the partition of the infinite dimensional FRF matrix measured in a test is identically equivalent to the inverse of a dynamically reduced impedance matrix. Again we see the presence of the OSET system dynamics in the term Z_{oo}^{-1}. The formula of the matrix inverse, given by equation (2.11) applies in the same fashion, and since every element in the Z_{oo}^{-1} is singular at the natural frequencies of the OSET, it is seen from equation (2.16) that the elements of H_{aa}^{-1} will be singular at the OSET natural frequencies (Gordis 1999).

Given that the measured FRF matrix implicitly defines a dynamically reduced impedance model, we pursue the reduction of the FEM in the same manner. That is, we calculate an *n* x *n* FRF matrix from the FEM, and simply extract the portion of this matrix which corresponds to the coordinates measured in a test. Hence the resulting extraction-reduced finite element FRF matrix retains all the modal content of the original model.

D. FREQUENCY RESPONSE FUNCTION [H]

The Frequency Response Function is a characteristic of the system where the frequency response is the frequency domain input-output relationship which can be considered the equivalent of the time domain *impulse response function* (Ewins, 2000). The FRF function contains the magnitude and phase (complex amplitude) of the response at the corresponding DOF "i" due to a harmonic force of unit amplitude at DOF "j". The time domain measured data is transformed to the frequency domain using the Fast Fourier Transform (FFT) algorithm.

The inverse of the FRF matrix $\left[H\right]^{-1}$ is known as the impedance matrix $\left[Z\right]$, that is:

$$\left[Z(\Omega)\right] = \left[H(\Omega)\right]^{-1} \tag{2.19}$$

and as it was implied from equations (2.5) and (2.6):

$$\left[Z(\Omega)\right] = \left[K - \Omega^2 M + j\Omega C\right] \tag{2.20}$$

Analyzing equation (2.6) in terms of an n DOF system

$$\begin{bmatrix} Z_{11} & \cdots & Z_{1n} \\ \vdots & \ddots & \vdots \\ Z_{n1} & \cdots & Z_{nn} \end{bmatrix} \begin{Bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.21}$$

Transferring from physical to modal coordinates let $\{x\} = [\Phi]\{q\}$, where Φ is the mass normalized mode shapes of the system and q represents the generalized coordinate set, and rewriting equation (2.21)

$$[Z][\Phi]\{q\} = \{f\} \tag{2.22}$$

Pre-multiply by $[\Phi]^T$ and expanding $[Z]$ from equation (2.20)

$$\left[[\Phi]^T[K][\Phi] - \Omega^2[\Phi]^T[M][\Phi] + j\Omega[\Phi]^T[C][\Phi]\right]\{q\} = [\Phi]^T\{F\} \tag{2.23}$$

Using orthogonality $[\Phi]^T[M][\Phi] = 1$, and assuming proportional damping $[C] = \alpha * [K] + \beta * [M]$:

$$\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]\{q\} = [\Phi]^T\{F\} \tag{2.24}$$

Where ω_i is the natural frequency of the i^{th} mode, ζ is the damping ratio and $\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]$ is known as the modal impedance matrix and is diagonal. This matrix is inverted to find the modal frequency response:

$$\{x\} = [\Phi]\frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]}[\Phi]^T\{F\} \tag{2.25}$$

Transforming back equation (2.25) into physical coordinates by pre-multiplying by $[\Phi]$ and using $\{\Im\} = [\Phi]^T\{F\}$, the modal decomposition of the FRF matrix in physical coordinates is as follows:

$$\{x\} = [\Phi]\frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]}[\Phi]^T\{F\} \tag{2.26}$$

where $[H(\Omega)]$ can be defined as:

$$[H(\Omega)] = [\Phi]\frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]}[\Phi]^T \tag{2.27}$$

Writing equation (2.27) in summation form:

$$[H(\Omega)] = \sum_{k=1}^{\text{modes}} \frac{\{\Phi^k\}\{\Phi^k\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{2.28}$$

Or for any element in terms of $\left[H_{ij}(\Omega)\right]$:

$$[H_{ij}(\Omega)] = \sum_{k=1}^{\text{modes}} \frac{\{\Phi_i^k\}\{\Phi_j^k\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{2.29}$$

E. DRIVING POINT AND TRANSFER POINT FUNCTIONS

The Frequency Response Function represents a harmonic response in the i^{th} coordinate, caused by a single harmonic force applied either at the same i^{th} coordinate or at a different j^{th} coordinate. Hence the Driving point Function is one where the response coordinate and the excitation coordinate are identical; where as the Transfer

Function is one where the response and excitation coordinates are located at different DOF points. The FRF is the foundation of the modal testing. It shows a direct connection between the modal properties of a system and its response characteristics; hence it provides an efficient means of predicting responses (Ewins, 1984).

The following example is a two DOF system with no damping (Ewins, 2000). It displays the characteristics of both a driving point and transfer function for the complete FRF.

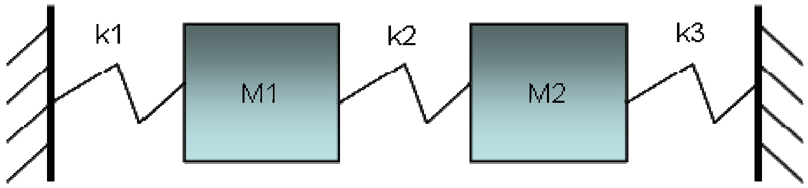

Figure 1. Two-DOF Example.

Assembling the stiffness matrix:

$$[K]=\begin{bmatrix} K_1+K_2 & -K_2 \\ -K_2 & K_2+K_3 \end{bmatrix}$$

The mass matrix is defined as:

$$[M]=\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$$

For the purpose of this example $M_1=M_2=1.0$ and $K_1 = K_3= 0.4$ and K_2=0.8. This yields the following results:

Mode shapes: $\{\Phi^1\}=\begin{Bmatrix}-0.7071\\-0.7071\end{Bmatrix}\ \{\Phi^2\}=\begin{Bmatrix}-0.7071\\0.7071\end{Bmatrix}$

Natural frequencies: $\{\omega(rad/\sec)\}=\begin{Bmatrix}0.6325\\1.0954\end{Bmatrix}$

Applying these values in equation $\left[H_{ij}(\Omega)\right] = \sum_{k=1}^{\mathrm{mod}\,es} \frac{\left\{\Phi_i^k\right\}\left\{\Phi_j^k\right\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k}$

For the Driving Point function

$$\left[H_{11}(\Omega)\right] = \frac{\left\{\Phi_1^1\right\}\left\{\Phi_1^1\right\}^T}{\omega_1^2 - \Omega^2} + \frac{\left\{\Phi_1^2\right\}\left\{\Phi_1^2\right\}^T}{\omega_2^2 - \Omega^2} = \frac{\{-0.7071\}\{-0.7071\}^T}{0.4 - \Omega^2} + \frac{\{-0.7071\}\{-0.7071\}^T}{1.2 - \Omega^2} \quad (2.30)$$

For the Transfer function.

$$\left[H_{12}(\Omega)\right] = \frac{\left\{\Phi_1^1\right\}\left\{\Phi_2^1\right\}^T}{\omega_1^2 - \Omega^2} + \frac{\left\{\Phi_1^2\right\}\left\{\Phi_2^2\right\}^T}{\omega_2^2 - \Omega^2} = \frac{\{-0.7071\}\{-0.7071\}^T}{0.4 - \Omega^2} + \frac{\{-0.7071\}\{0.7071\}^T}{1.2 - \Omega^2} \quad (2.31)$$

It is interesting to determine what controls whether a particular FRF will have positive or negative modal constants and thus whether it will exhibit antiresonances or not. As it was noticed from equation (2.29), it is the product of two eigenvectors elements, one at the response point and the other at the excitation point. Clearly, if we are to consider a driving point, then the modal constant for every mode must be positive, it being the square of a number. This means that for a point FRF, there must be an antiresonance following every resonance, without exception. This is displayed in the top portion of Figure 2 which is the Driving Point FRF$\left[H_{11}\right]$.

The situation for the Transfer Function is somewhat different because clearly the modal constant will sometimes be positive and sometimes negative, depending upon whether the excitation and response move in phase with each other or not. Thus we expect Transfer FRF measurements to show a mixture of antiresonances and minima. However, this mixture can be anticipated to some extent because it can be shown that, in general, the further apart are the two points in question, the more likely are the two eigenvector elements to alternate in sign as one progresses through the modes (Ewins, 2000).This is shown in the lower plot of Figure 2 which is the H_{12}. Conversely in the region between the two natural frequencies the two terms are additive and thus do not create an antiresonance.

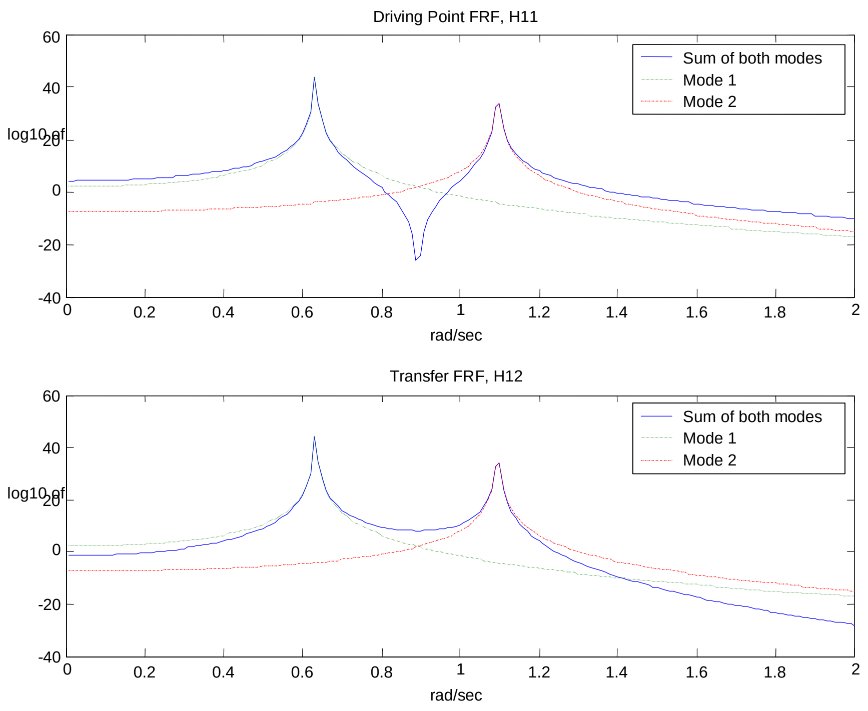

Figure 2. Two-DOF Frequency Response Function.

The principles demonstrated in the above example can be applied to any number of DOF. This concept of the existence of an antiresonance between any two modes of a driving point FRF is of great significance in the use of Artificial Boundary Conditions as it is shown in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ARTIFICIAL BOUNDARY CONDITIONS CONFIGURATION OF NATURAL FREQUENCIES

The improvement of a FEM is often a necessary step in order for the model to be used with confidence in the prediction of structural response. This improvement is difficult to achieve due to the spatially incompleteness of the measured FRF matrix giving an underdetermined problem. Undoubtedly there exists a recognized need to increase the size of the known parameter data base; it has been found that a number of additional mode frequencies can be identified from the same modal test without the need for any physical modification of the structure. These additional and distinct mode frequencies correspond exactly to those mode frequencies found when combinations of measured coordinates are restrained to ground. This last idea is referred as the ABC concept, which as it will be shown, improves the updating process by associating the natural frequencies with different boundary conditions.

A. RELATION BETWEEN THE DRIVING POINT ANTIRESONANCES TO ABC FRQUENCIES

Evoking the Driving Point Function concept discussed in the last chapter it can be shown that the ABC's frequencies of a given ASET coordinate are defined by the corresponding driving point antiresonances. Furthermore, the ABC's mode frequencies not only provide a greater number of frequencies for the system, but also provide a means to reduce ill-conditioning in the solution of the sensitivity equations.

From equation (2.29) the driving point FRF is

$$H_{ii}(\Omega) = \sum_{k=1}^{modes} \frac{(\Phi_i^k)^2}{\omega_k^2 - \Omega^2} \tag{3.1}$$

where Φ_i is the mass normalized mode shape element, ω_k is the ***k***th natural frequency, and Ω is the forcing frequency.

The frequency of the anti-resonance of $H_{11}(\Omega)$ is given by

$$\Omega^2_{anti-res} = \frac{R^1_{11}\omega^2_2 + R^2_{11}\omega^2_1}{R^1_{11} + R^2_{11}} \tag{3.2}$$

where the modal residue is giving by R_{ij}^k = {Φ(:, k)} {Φ(:, k)}. The following example (Gordis, 1999), shows how frequency is identical to the natural frequency of the system in Figure 3, to the driving point DOF constrained to ground as shown in Figure 4.

1. Calculation of ABC Frequencies for a 2-DOF System

Given the shown undamped system of Figure 3:

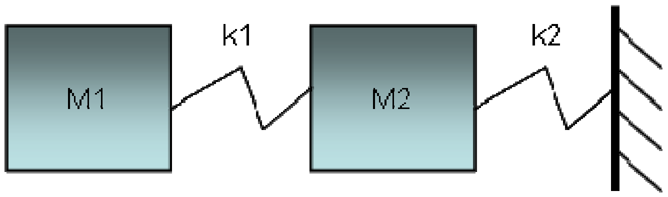

Figure 3. Two-DOF system.

Let $M_1 = M_2 = 1.0$ and $K_1 = K_2 = 1.0$, the frequency of the single antiresonance of the system is $\Omega_{antires} = \sqrt{2}$ rad/sec, which is identically equal to the single natural frequency of the ABC system of Figure 4 which $\omega = \sqrt{2}$ rad/sec.

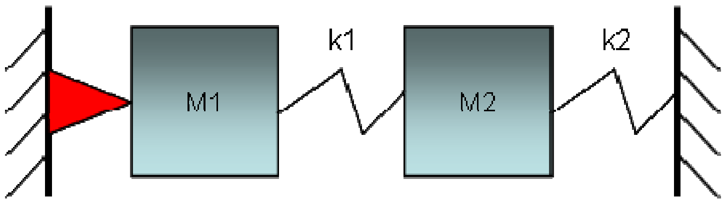

Figure 4. ASET, DOF #1 constrained to ground.

Mode shapes: $\left\{\Phi^{1}\right\}=\begin{Bmatrix}-0.85065\\-0.52573\end{Bmatrix} \qquad \left\{\Phi^{2}\right\}\begin{Bmatrix}-0.52573\\0.85065\end{Bmatrix}$

Natural frequencies: $\left\{\omega(rad/\sec)\right\}=\begin{Bmatrix}0.618\\1.618\end{Bmatrix}$

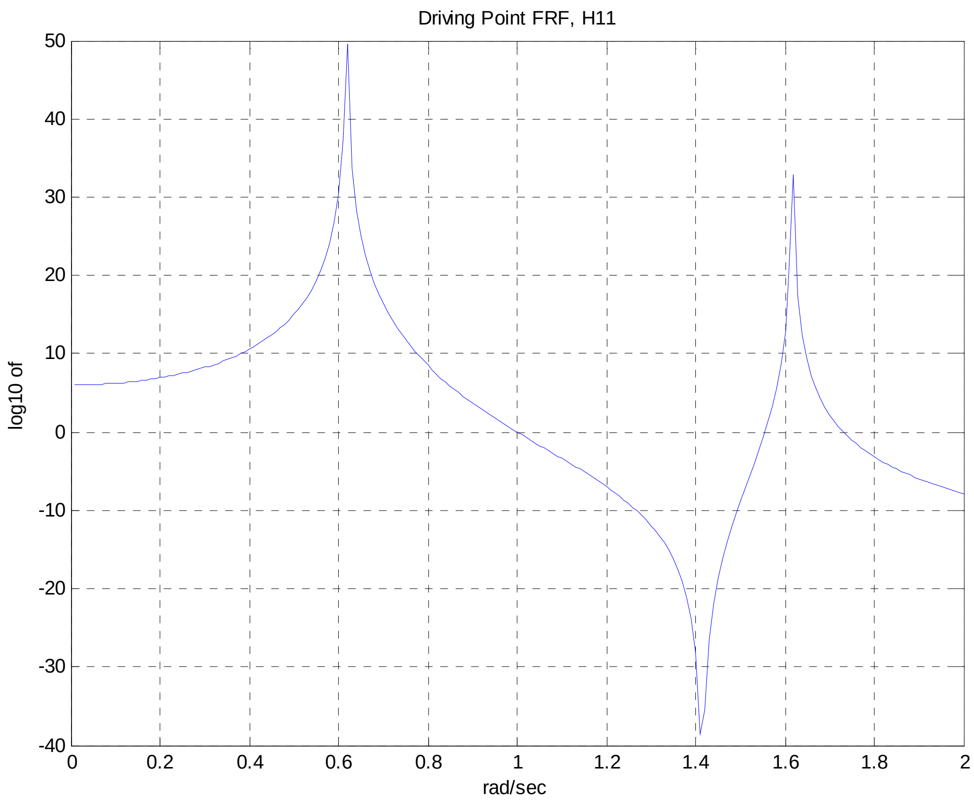

Figure 5. Driving Point FRF, H_{11}.

The $H_{11}(\Omega)$ plot in Figure 5 graphically shows the calculated mode frequencies, as well as the anti-resonance frequency as it was computed from equation (3.2) is $\Omega_{antires}=\sqrt{2}$ rad/sec. or 1.41 rad/sec.

By the use of equation (2.16), the frequencies of the ABC system can be found just by calculating inverse of $H_{11}(\Omega)$, where the ASET is pinned at DOF #1, and $H_{aa}^{-1}(\Omega)$

is the inverse of $H_{11}(\Omega)$. A clear correlation is depicted in the bottom graph of Figure 6, where the single natural frequency corresponds to the antiresonance frequency. Furthermore, this singular frequency happens to be the natural frequency of the ABC system obtained by constraining DOF #1 to ground (Figure 4). Hence it is shown the power and usefulness of equation (2.16), where the driving point antiresonance correspond directly to the natural frequencies of the system, with the driving point coordinate constrained to ground (Gordis, 1999).

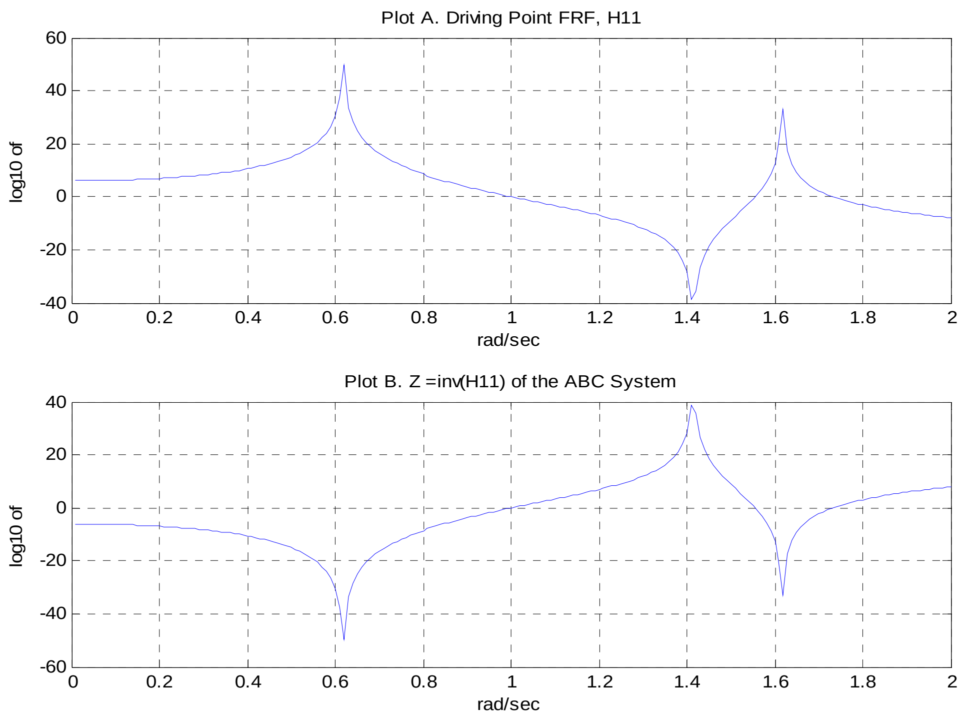

Figure 6. Plot A. Driving Point H11(Ω) of system 1, Plot B. $[H_{aa}(\Omega)]^{-1}$ of system 2.

2. Calculation of ABC Frequencies for a Free-Free Beam

To further show the usefulness of the ABC concept, a free-free beam is analyzed (Gordis, 1999). The present model depicted in Figure 7 is examined, it consists of 10 beam elements and each element contains two nodes with two DOF each, giving a total of 22 DOF system. The translational DOF correspond to the odd numbers where as the rotational DOF correspond to the even numbers. Response transducers are set up at the translational DOF: 1, 5, 9, 13, 17, and 21 as follows:

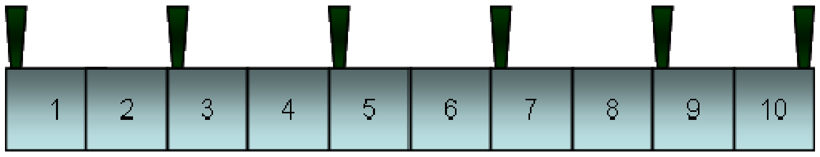

Figure 7. Transducers located at DOF 1, 5, 9, 13, 17, and 21.

The set of coordinates chosen to be measured with response accelerators represent the ASET, defined as [1 5 9 13 17 21]. It can be assumed that the excitation is applied at each of the ASET DOF, resulting in a 6 x 6 FRF matrix. The impedance matrix $H_{aa}^{-1}(\Omega)$, the scalar inverse of $H_{11}(\Omega)$, is calculated at excitation frequencies from 0-800 hertz. Figure 8 shows the driving point FRF of the system.

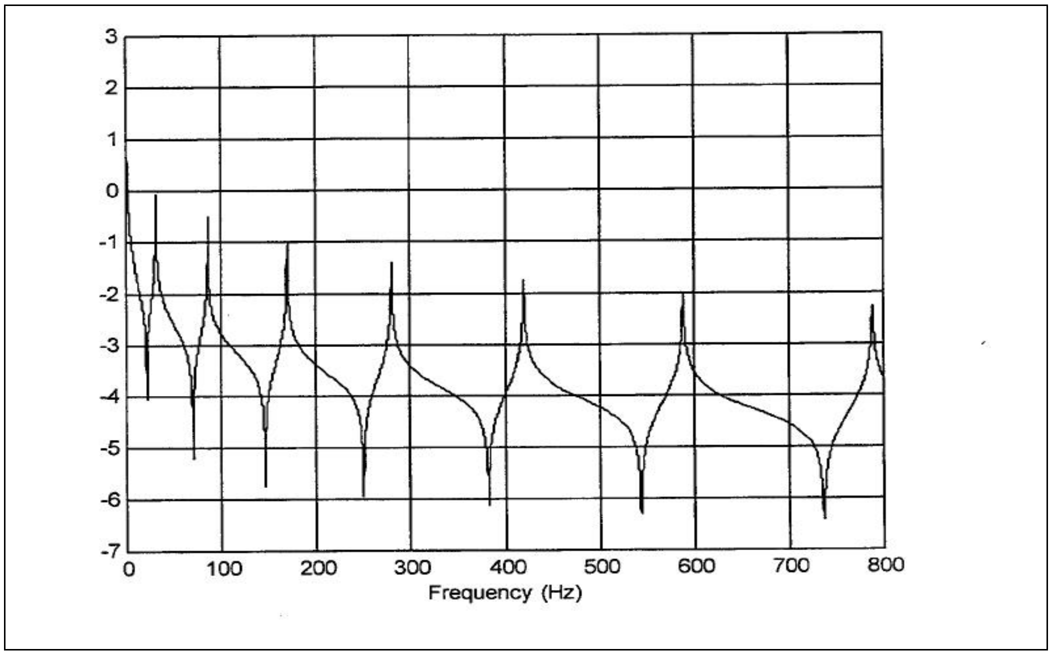

Figure 8. Driving Point FRF ASET DOF: 1 5 9 13 17 21. [From Gordis, 1999].

The ABC system frequencies are found by calculating the impedance matrix $H_{aa}^{-1}(\Omega)$. By plotting the 1,1 element $H_{11}(\Omega)$ of the matrix, the ABC system frequencies can be seen in Figure 9:

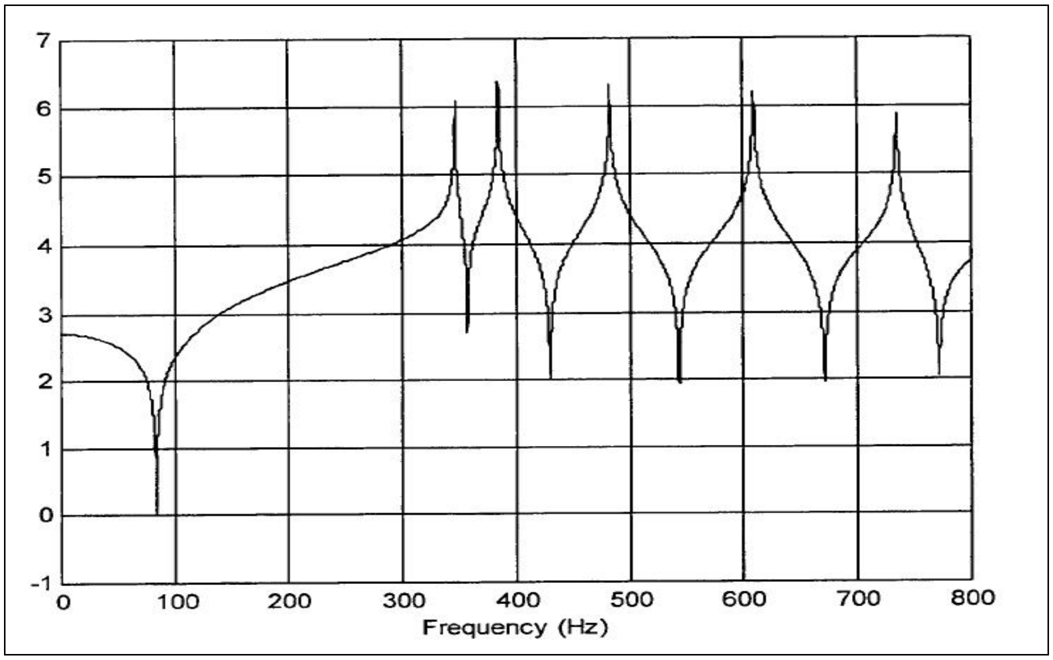

Figure 9. Impedance FRF for ASET DOF: 1 5 9 13 17 21. [From Gordis, 1999].

The ABC frequencies correspond exactly to the natural frequencies of the system with all of the measured coordinates ASET constrained to ground. This configuration is shown in Figure 10:

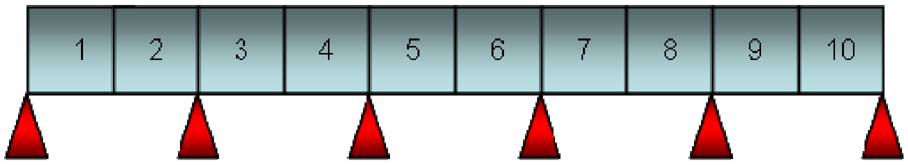

Figure 10. ABC Configuration ASET [1 5 9 13 17 21] (Measured coordinates restrained to ground).

IV. SENSITIVITY BASED FEM UPDATING USING ABC

The disparity in the number of known parameters (measured) versus the number of parameters to be adjusted in order to update a FEM, defines a spatially incomplete structure as well as an underdetermined problem.

The importance of obtaining sensitivities of the dynamic characteristics of a system with respect to the changes of its parameters lies in the fact that they are critical for achieving efficient design modification, and for predicting the optimal structural modifications within a prescribed dynamic characteristics change. In such a case, there will be numerous possible modifications that can accomplish the change. Sensitivity analysis can provide the most effective answer.

From a mathematical point of view, sensitivity analysis implies differentiation. It can be carried out on the modal data from the FRF of a dynamic system. The variables in this analysis are usually system parameters such as mass or stiffness parameters. The functions of the analysis can be natural frequencies and mode shapes of the system (modal data), or its FRF. The assumption of infinitesimal changes demanded by differentiation applies to sensitivity analysis. Hence the sensitivity based updating concept is used for error localization in order to find the parameters requiring adjustments. The ABC concept can be used in addition to the standard baseline system mode frequencies in model updating and damage detection. As stated previously, the ABC frequencies correspond to the same structural system as the baseline frequencies, but with different boundary conditions. The governing equation for sensitivity based updating is:

$$\left\{\Delta\omega^2\right\} = \left[T\right]\left\{\Delta Dv\right\} \tag{4.1}$$

where $\left\{\Delta\omega^2\right\}$ is a vector of natural frequency errors. The errors are the difference between the experimental and the analytical natural frequencies $\{\omega^x - \omega^a\}$. The $\left\{\Delta Dv\right\}$

term is the vector of changes to be calculated for the specified model parameters, known as the design variables, and $\left[T\right]$ is the sensitivity matrix.

Each column of the sensitivity matrix represents an element of the model, where as each row represents a mode of the model.

Each ABC system defines additional rows to equation (4.1) that is it adds more modes, defining the equation:

$$\left\{\Delta\omega^{2(i)}\right\} = \left[T^{(i)}\right]\left\{\Delta Dv\right\} \tag{4.2}$$

where $T_{ij}^{(k)} = \dfrac{\partial \omega_i^{2(k)}}{\partial DV_j}$ and $\omega_i^{2(k)}$ represents the *i*th natural frequency of the *k*th ABC configuration system. The baseline system quantities are represented with the superscript "0" and the ABC systems are identified with the superscripts from 1 to "*k*", where *k* represents the number of ABC used in the system. Combining baseline and ABC system equations:

$$\begin{Bmatrix} \Delta\omega_{BASE}^{2(0)} \\ \Delta\omega_{ABC1}^{2(1)} \\ \vdots \\ \Delta\omega_{ABCk}^{2(k)} \end{Bmatrix} = \begin{pmatrix} T^{(0)}{}_{BASE} \\ T^{(1)}{}_{ABC1} \\ \vdots \\ T^{(k)}{}_{ABCk} \end{pmatrix} \begin{Bmatrix} dV_1 \\ \vdots \\ dV_n \end{Bmatrix} \tag{4.3}$$

The degree of coupling between the ABC systems and the baseline system in equation (4.3) can be adjusted by deleting or retaining individual columns of the $\left[T^k\right]$. Partial coupling between the baseline system and the ABC system is established if the design variables (DV) are partitioned. That is, some of the DV are associated only with the baseline system, $\{\Delta DV^0\}$, some are associated with both the baseline and the ABC system, $\{\Delta DV^{0,1}\}$, and some are associated only with the ABC system, $\{\Delta DV^1\}$, (Gordis, 1999).The use of the ABC system sensitivities helps to eliminate or at least to reduce the problem of poorly conditioned or rank-deficient [T] matrices. Columns of $\left[T^0\right]$ can be replaced with columns of $\left[T^k\right]$ in order to improve the conditioning. Two closely spaced elements in a model will have nearly dependent columns of $\left[T^0\right]$, preventing the discrimination of error or damage between two elements. One column of $\left[T^0\right]$ associated with one of the two elements can be replaced with a column of $\left[T^k\right]$, and the associated baseline system natural frequency replaced with the ABC system frequency (Gordis, 1999).

A. SENSITIVITY MATRIX DERIVATION [T]

Several approaches are available for performing a sensitivity analysis. The following derivation is based on the parameterization of the eigenvalue problem (Inman, 1989). Consider an *n* DOF system defined by:

$$M(DV)\ddot{x}(t) + K(DV)x(t) = 0 \tag{4.4}$$

where the corresponding eigenvalues problem is:

$$\left[K - \lambda_i M\right]\left\{\Phi_i\right\} = \left\{0\right\} \tag{4.5}$$

Differentiating equation (4.5) with respect to the design variable {DV}, as it is the vector of design parameters representing the change in mass matrix $\left[M\right]$ and/or change in stiffness matrix $\left[K\right]$, which are adjustable for the FE model.

$$\left[\frac{\Delta K}{\Delta DV} - \lambda_i \frac{\Delta M}{\Delta DV} - \frac{\Delta \lambda_i}{\Delta DV}\right]\left\{\Phi_i\right\} + \left[K - \lambda_i M\right]\left\{\frac{\Delta \Phi_i}{\Delta DV}\right\} = \left\{0\right\} \tag{4.6}$$

Expanding equation (4.6) and pre-multiplying by $\left\{\Phi\right\}^T$ leads to:

$$\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}\right]\left\{\Phi_i\right\} - \lambda_i\left\{\Phi_i\right\}^T\left[\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\} - \left[\frac{\Delta \lambda_i}{\Delta DV}\right]\left\{\Phi_i\right\}^T\left[M\right]\left\{\Phi_i\right\} + \left\{\Phi_i\right\}^T\left[K - \lambda_i M\right]\left\{\frac{\Delta \Phi_i}{\Delta DV}\right\} = \left\{0\right\} \tag{4.7}$$

And invoking the matrix identity property $\left\{a\right\}^T\left[b\right]\left\{c\right\} = \left\{c\right\}^T\left[b\right]\left\{a\right\}$ the last term on the left hand side can be replaced by:

$$\left\{\frac{\Delta \Phi_i}{\Delta DV}\right\}^T\left[K - \lambda_i M\right]\left\{\Phi_i\right\} = 0 \tag{4.8}$$

Since $\left[K-\lambda_i M\right]\left\{\Phi_i\right\}=\left\{0\right\}$ the overall equation is reduced

$$\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}\right]\left\{\Phi_i\right\}-\lambda_i\left\{\Phi_i\right\}^T\left[\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\}-\left[\frac{\Delta\lambda_i}{\Delta DV}\right]\left\{\Phi_i\right\}^T\left[M\right]\left\{\Phi_i\right\}=\left\{0\right\} \tag{4.9}$$

Using orthogonality, where $\left[\Phi_i\right]^T\left[M\right]\left[\Phi_i\right]=1$ equation (4.9) further reduces to

$$\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}\right]\left\{\Phi_i\right\}-\lambda_i\left\{\Phi_i\right\}^T\left[\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\}-\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\left\{0\right\} \tag{4.10}$$

Bringing the mass and stiffness portion to the right hand side

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}-\lambda_i\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\} \tag{4.11}$$

Finally it is demonstrated that the sensitivity matrix involves differentiation with respect to the design parameter where as for this case are either consider mass and/or stiffness, and the function of the analysis is with respect to the natural frequencies of the system as it is shown in equations (4.12) and (4.14):

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}\right]\left\{\Phi_i\right\} \quad \text{where} \quad \left[\Delta K\right]=\left[K_x-K_a\right] \tag{4.12}$$

and

$$\left[T_{stiffness}\right]=\left\{\Phi_i\right\}^T\left[\frac{\Delta K}{\Delta DV}\right]\left\{\Phi_i\right\} \tag{4.13}$$

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\left\{\Phi_i\right\}^T\left[-\lambda_i\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\} \quad \text{where } \left[\Delta M\right]=\left[M_x-M_a\right] \tag{4.14}$$

and

$$\left[T_{mass}\right]=\left\{\Phi_i\right\}^T\left[-\lambda_i\frac{\Delta M}{\Delta DV}\right]\left\{\Phi_i\right\} \tag{4.15}$$

B. BEAM SPECIFICATIONS

A Finite Element computer program was written to calculate the Sensitivity matrix of a twenty element cantilever beam model, which consists of an aluminum bar, with a partially drilled and tapped 10/32" diameter hole at the free end, which allows the load cell to be attached to the structure. Table 1 provides a summary of the beam specifications.

PARAMETER	VALUE
Length	42 inches
Width	1.5 inches
Thickness	0.5 inches
Density	$0.11\, lb / in^3$
Elasticity Modulus	10 E6 $lbf\ / \sec^2 - in$.

Table 1. Beam Specifications.

C. MASS AND STIFFNESS SENSITIVITY ANALYSIS

The computer program built to assemble the sensitivity matrix calculates the mode frequency sensitivities according to equation (4.11). This process is performed in two steps. The first step calculates the Mass Sensitivity matrix and the second calculates the Stiffness Sensitivity matrix. Hence the [T] matrix is of size m x p, m being the number of modes and p being the number of elements.

In the Mass Sensitivity calculation, a small perturbation of 1% of mass was applied to each element of the beam; this is done using equation (4.14). Each subsequent column of [T] was calculated at every element of the model.

On step two for the Stiffness Sensitivity matrix calculation, the same 1% stiffness perturbation is applied iteratively to each element and equation (4.12) is used to assemble the corresponding part of the Sensitivity matrix.

Once the two steps are completed, both sensitivities matrices are assembled [T(M)] & [T(K)], hence obtaining a [T] total of size m x 2p, where the first set of twenty columns yield the changes in mass and the second set o twenty columns yields the changes in stiffness.

The following plots show a normalized mass and stiffness sensitivity distribution along the 20 elements of the beam model. Each element is indicated by a bar which represents its corresponding response to the change of the system natural frequency, with respect to the corresponding change of the design variable.

It will be seen in an overall sense on all of the Figures below how significantly the sensitivity is influenced by the mode shapes. It is clear from Figure 11, how the Base Stiffness Sensitivity is more likely to detect a change in the natural frequency closer to the fixed end of the cantilever beam model, rather than at the free end, where the red sensitivity bars are almost zero.

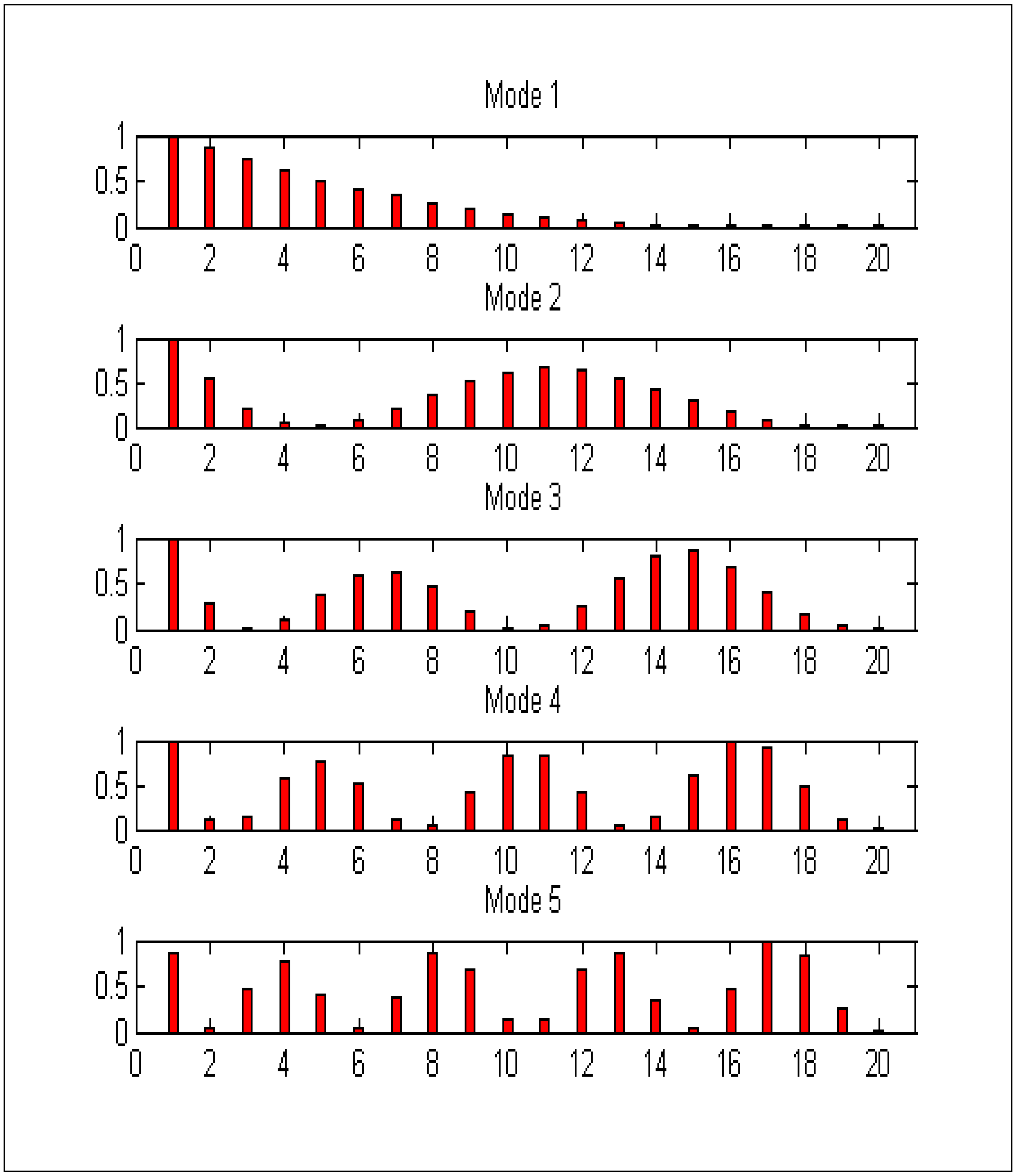

Figure 11. Stiffness Base Sensitivity Matrix. Modes 1 to 5.

In Figure 12, it is evident how the Base Mass Sensitivity is more likely to detect a change in the natural frequency, closer to the free end of the cantilever beam model, rather than at the fixed end, where the magenta sensitivity bars are almost zero.

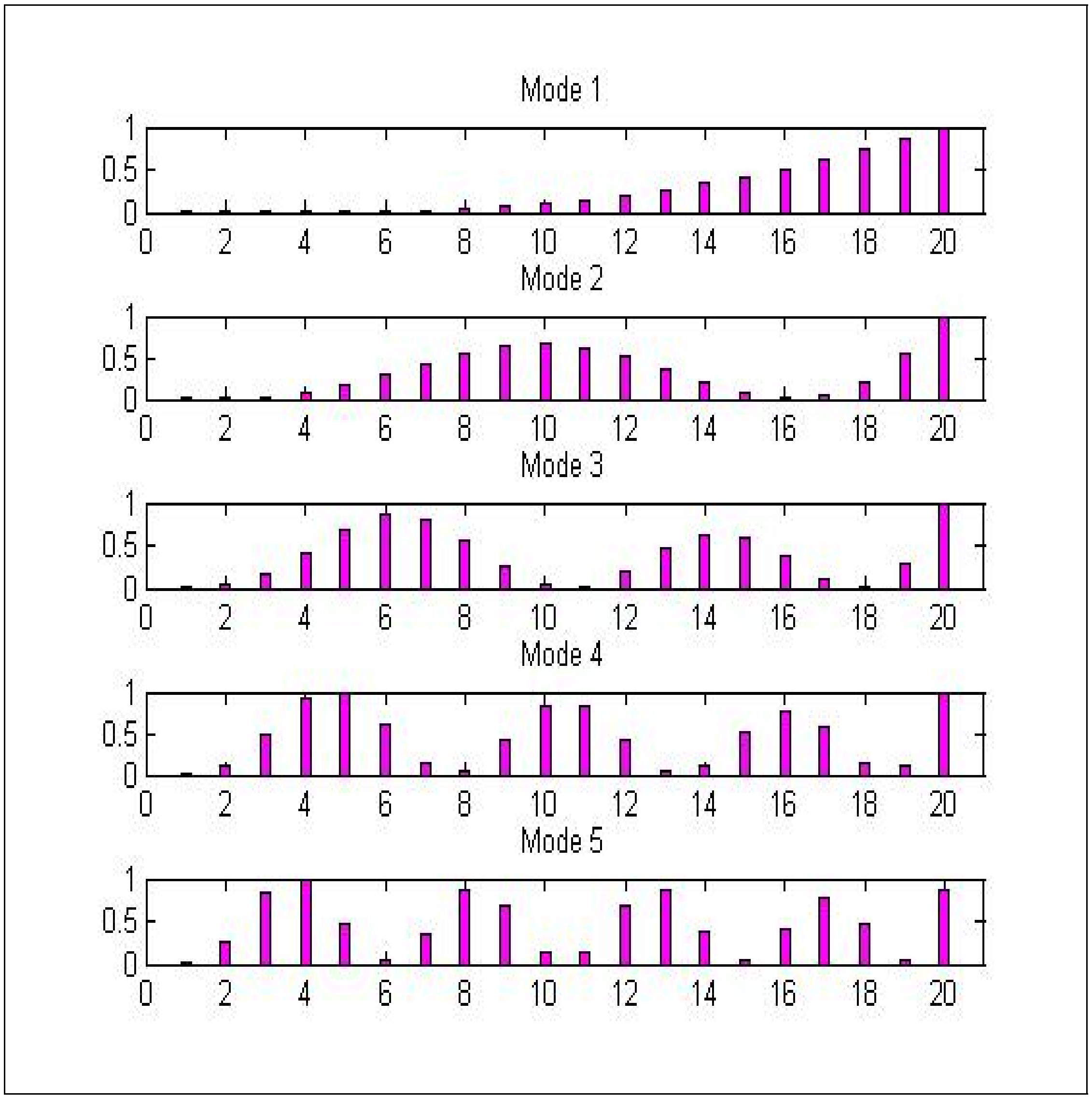

Figure 12. Mass Base Sensitivity Matrix. Modes 1 to 5.

It should be noted that the Mass Sensitivity rows are represented in absolute values, therefore an increase of mass would generally lower the system natural frequencies, but, it was preferred to show the influence with respect to element position and hence all magnitudes were made positive.

The next four figures show a particular trend followed by the Stiffness and Mass Sensitivities when ABC's are applied to the system.

The green pointer with triangular shape at element twelve of Figures 13 and 15 and element four of Figures 14 and 16, represent the pinned ABC applied at DOF 25 and 9, respectively as follows:

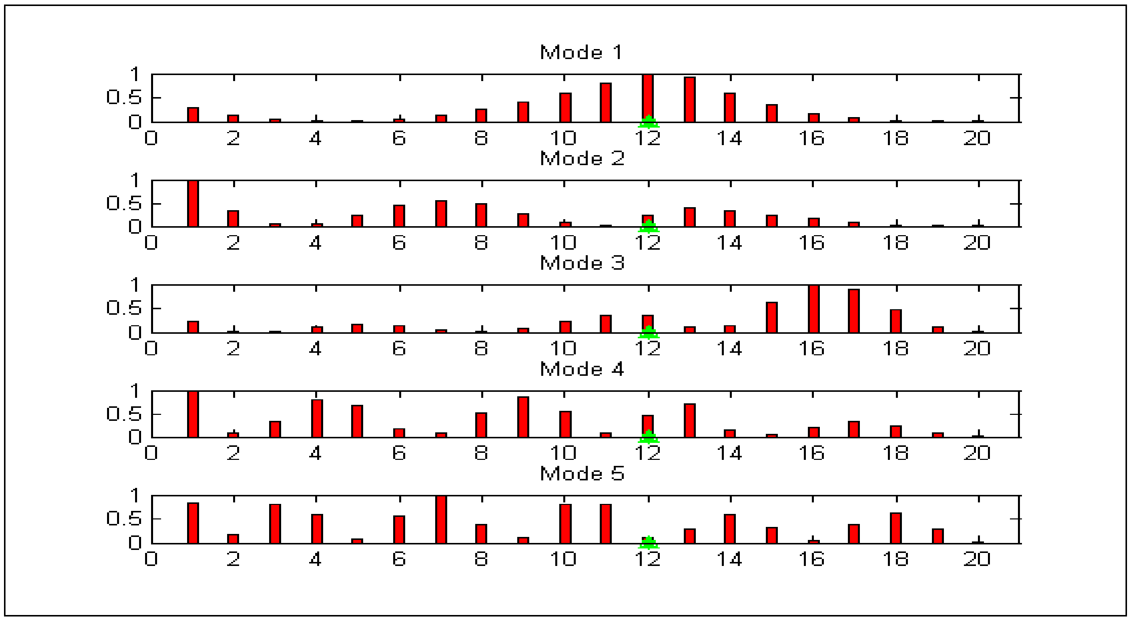

Figure 13. Stiffness Sensitivity distribution with ABC at DOF 25.

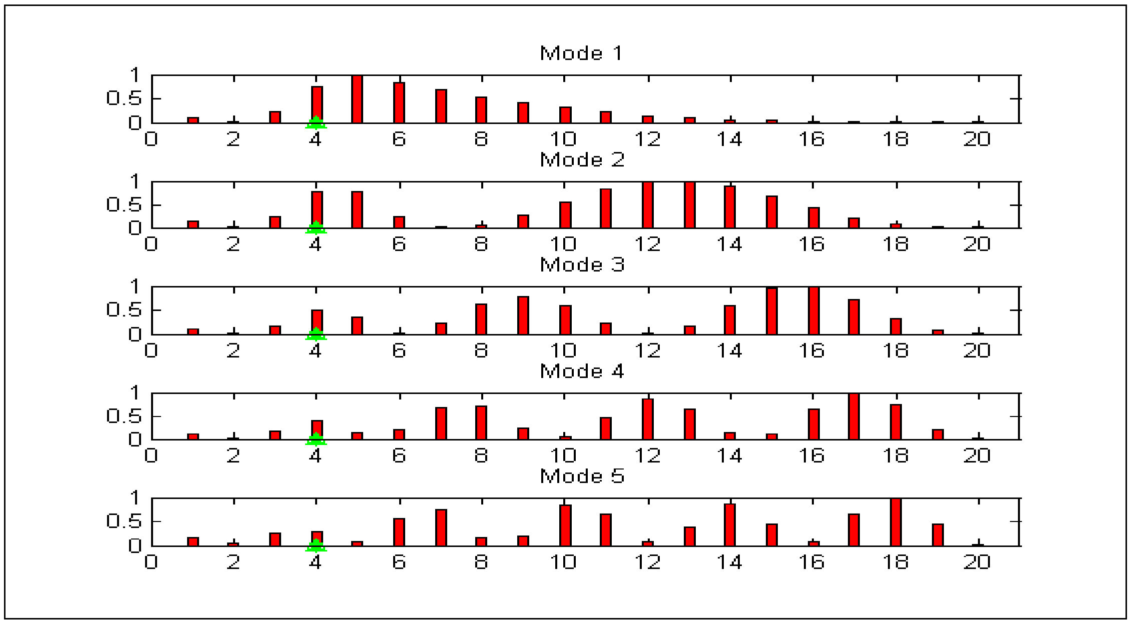

Figure 14. Stiffness Sensitivity distribution with ABC at DOF 9.

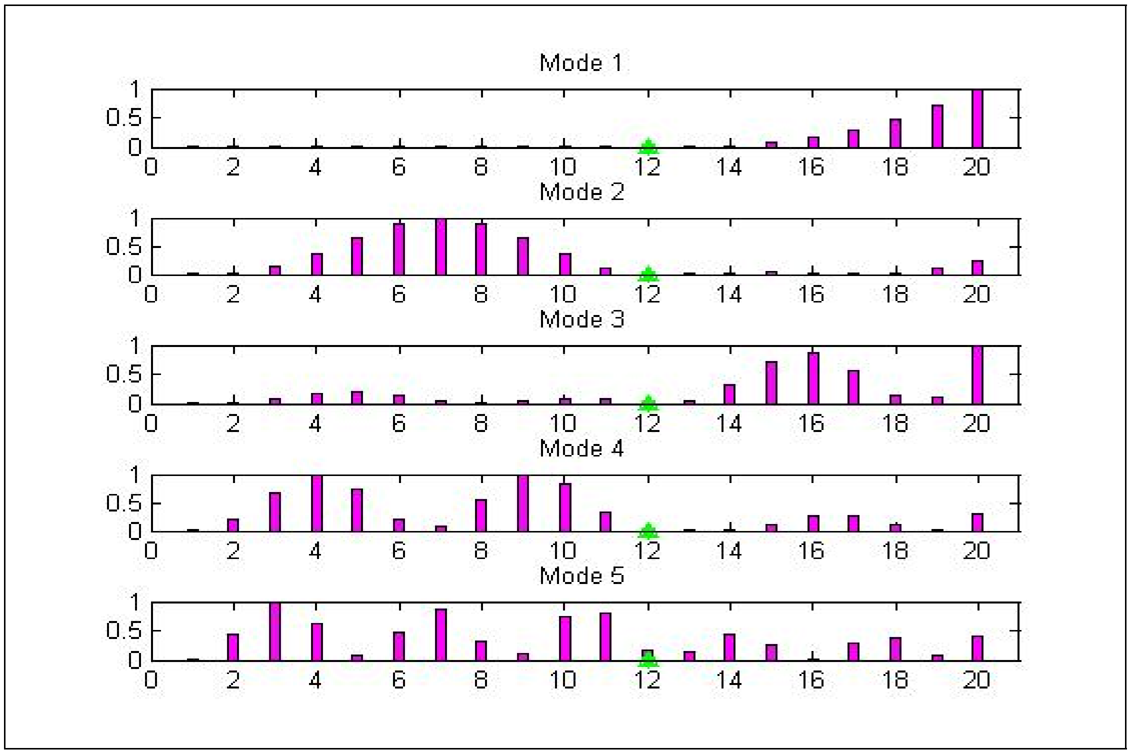

Figure 15. Mass Sensitivity distribution with ABC at DOF 25.

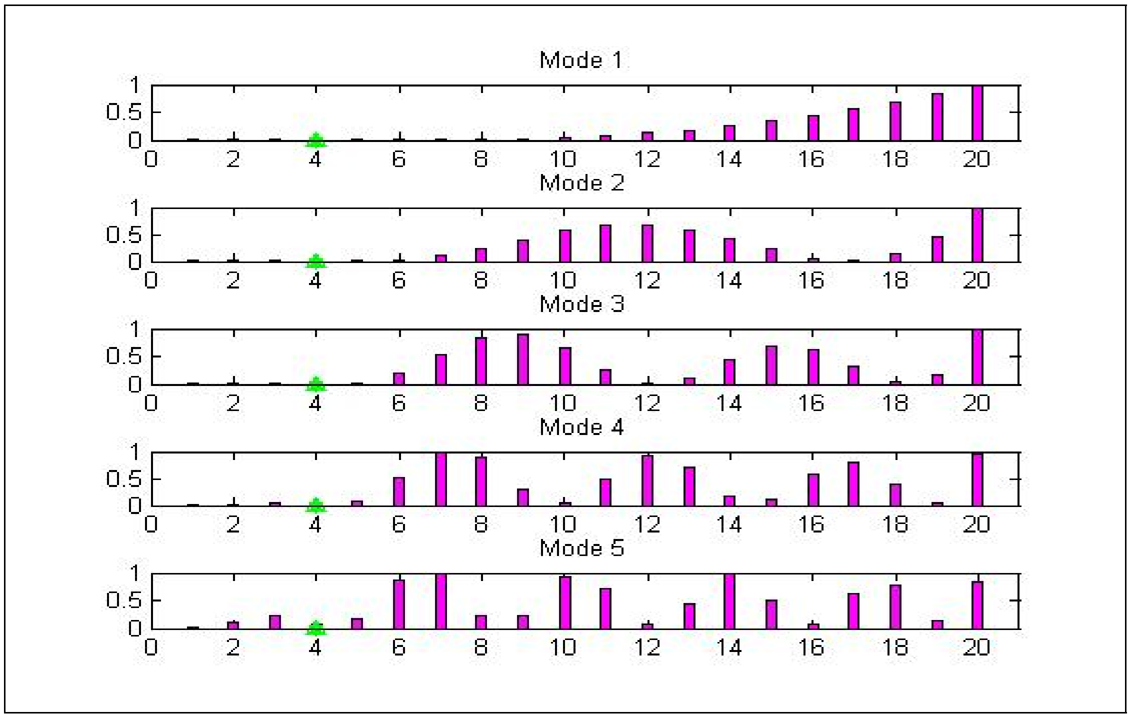

Figure 16. Mass Sensitivity distribution with ABC at DOF 9.

In general when ABC's are applied to the Stiffness Sensitivity matrix, the model has a tendency to increase the values of their sensitivities at that particular "pinned" ABC, whereas for the Mass Sensitivity, the pinned ABC seems to drive the sensitivity values away from the DOF's where fixed conditions are emulated; these fixed conditions can be thought as those representing the fixed end of the beam as well as the pinned ABC set up at the corresponding DOF's 25 and 9 as it was showed on Figures 15 and 16, respectively.

In summary, the previous Figures expressed that in places of low sensitivity, the beam model is not very likely to predict the changes in natural frequencies due to the change in the design variable, and that is, in a real world scenario those low sensitivity areas would be the places where damage, if any, is not going to be accurately detected. Hence, the sensitivity concept leads to a key role in the updating of a FEM. Furthermore, the sensitivity analysis showed us two important points. The first is the big influence of the mode shapes into the sensitivity distribution of the model, and two; how the correct manipulation of the ABC's especially with regard to the stiffness sensitivity portion, could lead to a better prediction of the errors in the FEM updating.

THIS PAGE INTENTIONALLY LEFT BLANK

V. KINETIC AND STRAIN ENERGY CORRELATION TO SENSITIVITY MATRIX

Because the natural frequencies and mode shapes of a structure are dependent on the mass and stiffness distributions, any subsequent changes in them should, theoretically, be reflected in changes in the frequency and mode shapes of the structure. Hence, the first step in a vibration problem is to determine the important natural frequencies of the system; there are two general methods to determine the natural frequencies of the system: The Newtonian and the Energy method approach.

The Rayleigh's Energy method employs an energy balance which is essentially the scope of this chapter. For the sake of the following development, the Euler-Bernoulli beam theory is utilized, where the Euler-Bernoulli equation for beam bending is:

$$\varphi \frac{\partial^2 v}{\partial t^2} + \frac{\partial^2}{\partial x^2}\left(EI \frac{\partial^2 v}{\partial x^2} \right) = q(x,t) \tag{5.1}$$

where $v(x,t)$ is the transverse displacement of the beam, φ is the mass density per length, EI is the beam rigidity, $q(x,t)$ is the externally applied pressure loading, t and x indicate time and spatial axis along the beam axis. Rearranging equation (5.1) to develop the finite element formulation and the corresponding matrix equations, the averaged weighted residual of equation (5.1) is:

$$I = \int_0^L \left(\varphi \frac{\partial^2 v}{\partial t^2} + \frac{\partial^2}{\partial x^2}\left(EI \frac{\partial^2 v}{\partial x^2} \right) - q \right) \omega dx = 0 \tag{5.2}$$

where L is the length of the beam and ω is a test function. The weak formulation of equation (5.2) is obtained from integration by parts twice for the second term of the equation. In addition, discretization of the beam into a number of finite elements gives:

$$I = \sum_{i=1}^{n} \left(\int_{\Omega^e} \varphi \frac{\partial^2 v}{\partial t^2} \omega dx + \int_{\Omega^e} EI \frac{\partial^2 v}{\partial x^2} \frac{\partial^2 \omega}{\partial x^2} dx - \int_{\Omega^e} q \omega dx \right) + \left(-V\omega - M \frac{\partial \omega}{\partial x} \right)_0^L = 0 \tag{5.3}$$

where the slope is represented by:

$$\theta = \left(\frac{\partial v}{\partial x} \right) \tag{5.4}$$

Bending moment:

$$M = EI \left(\frac{\partial^2 v}{\partial x^2} \right) \tag{5.5}$$

Shear force:

$$V = -EI \left(\frac{\partial^3 v}{\partial x^3} \right) \tag{5.6}$$

and the Loading:

$$w = \left(\frac{\partial^4 v}{\partial x^4} \right) \tag{5.7}$$

Ω^e represents the element domain and n is the number of elements for the beam.

Shape functions for the spatial interpolation of the transverse deflection v in terms of nodal variables are considered. Elements are measured as having two nodes, one at each end as shown in Figure 17. The deformation of a beam must have continuous slope as well as continuous deflection at any two neighboring beam elements. To satisfy this continuity requirement each node has both deflection, v_i and slope θ_i as nodal variables. In this case any two neighboring beam elements have common deflection and slope at the shared nodal point. This satisfies the continuity of both deflection and slope. The Euler-Bernoulli beam equation is based on the assumption that the plane normal to the neutral axis before deformation remains normal to the neutral axis after deformation. This assumption denotes $\theta = \left(\frac{\partial v}{\partial x} \right)$ (i.e. slope is the first derivative of deflection in terms of x). (Kwon and Bang, 2000).

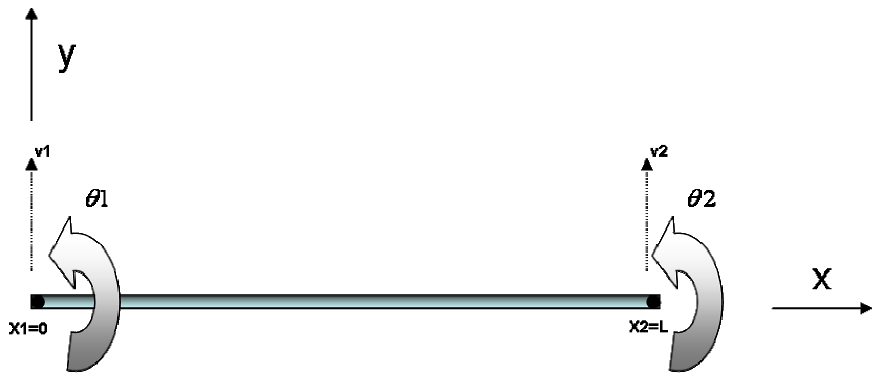

Figure 17. Two-noded beam element. [After Kwon and Bang, 2000].

Because there are four nodal variables for the beam element, we assume a cubic polynomial function for $v(x)$:

$$v(x) = c_o + c_1 x + c_2 x^2 + c_3 x^3 \tag{5.8}$$

The slope is computed from equation (5.8):

$$\theta(x) = c_1 + 2c_2 x + 3c_3 x^2 \tag{5.9}$$

Evaluation of the deflections and slopes at the nodes yields:

$$\begin{aligned} v(0) &= c_0 = v_1 \\ \theta(0) &= c_1 = \theta_1 \\ v(l) &= c_0 + c_1 l + c_2 l^2 + c_3 l^3 = v_2 \\ \theta(l) &= c_1 + 2c_2 l + 3c_3 l^2 = \theta_2 \end{aligned} \tag{5.10}$$

where l represents the element length. Solving equation (5.10) for c_i in terms of the nodal variables v_i and θ_i, substituting the results into equation (5.8) gives:

$$v(x,t) = H_1(x)v_1(t) + H_2(x)\theta_1(t) + H_3(x)v_2(t) + H_4(x)\theta_2(t) \tag{5.11}$$

where $H_i(x)$ represents the Hermitian shape functions, which are of C^1 type, meaning that both, v and $\frac{dv}{dx}$ are continuous between the neighboring elements.

In general, C^n type continuity means the shape functions have continuity up to the n^{th} order derivative between two neighboring elements. The Hermitian Shape functions for the two node beam element depicted in Figure 17 are:

$$\begin{aligned}
H_1(x) &= 1 - \frac{3x^2}{l^2} + \frac{2x^3}{l^3} \\
H_2(x) &= x - \frac{2x^2}{l} + \frac{x^3}{l^2} \\
H_3(x) &= \frac{3x^2}{l^2} - \frac{2x^3}{l^3} \\
H_4(x) &= -\frac{x^2}{l} + \frac{x^3}{l^2}
\end{aligned} \tag{5.12}$$

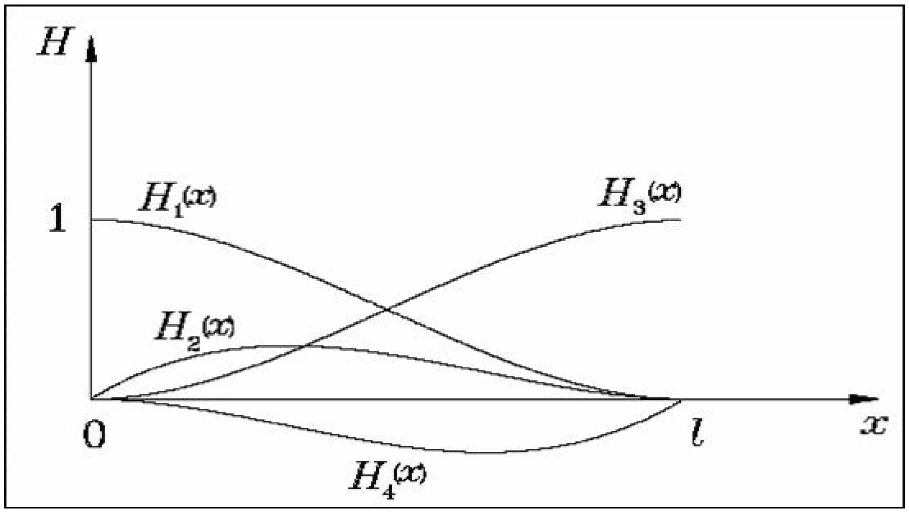

Figure 18. Hermitian beam element. [From Kwon and Bang, 2000].

A. STRAIN AND KINETIC ENERGY DERIVATION

Regarding the Strain Energy due to bending, consider a differential element of beam of length dx, where the axial stress due to bending is:

$$\sigma = \frac{Mz}{I} \tag{5.13}$$

where I is the cross section area of inertia, the Strain Energy per unit volume is:

$$dw = dU = \frac{1}{2} F dl \tag{5.14}$$

where dl is the change in length of a fiber of the beam segment *dx*. The work done by the applied moment induces axial strain in the fiber, which is stored as potential (strain energy).

Let $F = \sigma dA$, and $dL = \frac{dL}{L} L = \varepsilon dx$, then the Srain Energy becomes:

$$dU = \frac{1}{2}(\sigma dA)(\varepsilon dx) = \frac{1}{2}\sigma\varepsilon dAdx = \frac{1}{2}\sigma\varepsilon dVol \tag{5.15}$$

Let $dVol = 1$, then equation (5.15) becomes:

$$dU = \frac{1}{2}\sigma\varepsilon dVol = \frac{1}{2}\sigma\varepsilon \tag{5.16}$$

Since $\varepsilon = \frac{Mz}{EI}$ and from equation (5.13), the Strain Energy per unit volume becomes:

$$dU = \frac{1}{2E}\left(\frac{Mz}{I}\right)^2 \tag{5.17}$$

The total Strain Energy is:

$$U = \int_{vol} dU = \int_0^L \int_A \frac{1}{2E}\left(\frac{Mz}{I}\right)^2 dAdx \tag{5.18}$$

Using equation (5.5) and $I = \int_A z^2 dA$, the Strain Energy stored in a beam of uniform cross section that has a deflected elastic curve *v(x)* is:

$$U = \frac{EI}{2}\int_0^L \left(\frac{\partial^2 v(x,t)}{\partial x^2}\right)^2 dx \tag{5.19}$$

With respect to the Kinetic Energy, let the displacement for any cross section along the beam, to vibrate in a single mode at its corresponding natural frequency ω_n:

$$v(x,t) = H_i(x) * C_n \sin(\omega_n t + \theta_n) \tag{5.20}$$

Where $C_n \sin(\omega_n t + \theta_n)$ represents the temporal nodal coordinates comprised of displacements $v_i(t)$ or slopes $\theta_i(t)$ as it is implied from equation (5.11). The velocity of the cross section becomes:

$$\dot{v}(x,t) = H_i(x) * \omega_n C_n \cos(\omega_n t + \theta_n) \tag{5.21}$$

Let $C_n = 1$, then the maximum velocity is:

$$\dot{v}_{\max}(x) = H_i(x) * \omega_n \tag{5.22}$$

Notice that $\dot{v}_{\max}(x)$ becomes a function of x only. And the Total Kinetic Energy for the beam is (mass/length γ):

$$T_{\max} = \frac{1}{2} m v_{\max}^2 = \frac{\gamma}{2} \int_0^L \left(H_i(x) * \omega_n \right)^2 dx \tag{5.23}$$

Equations (5.19) and (5.23) were used to derive the Strain and Kinetic Energy Sensitivities, respectively in the twenty element beam which characteristics are described on Table 1. In order to derive the Strain Energies, in accordance with equation (5.19), the second derivatives of the Hermitian shape functions were multiplied by the corresponding displacement v_i or rotation θ_i of the eigenvector matrix (mode shapes), generated from the Build2beams.m program in MATLAB. Once the iterative process is completed for the number of elements, the Strain Energy matrix is assembled and it is of size *m* x *p*, where the *m* rows represent the displacement and rotations and the *p* columns represent the different mode shapes, as opposed to the Sensitivity matrix where the *m* rows represent the mode shapes and the *p* columns represent the different elements of the model.

In regard to the Kinetic Energy calculation, it is implied from equation (5.23), no modifications were performed to the Hermitian shape functions which are multiplied by the natural frequencies generated in the Build2beams.m MATLAB program. After the iterative process is completed, the Kinetic Energy matrix is assembled with the same characteristics of the Strain Energy matrix, that is the *m* rows representing the displacements and/or rotations and the *p* columns representing the different mode shapes.

The following graphs show the normalized plots for Strain and Kinetic Energies distribution, in the same style as the Sensitivity distribution figures shown in the previous chapter.

Again each subplot corresponds to a specific mode shape and each bar represents the level of either Strain or Kinetic Energy stored at each element along the corresponding mode shape. The red bars on Figure 19 depict the Strain energy distribution along the twenty elements Base system of the beam over the first five mode shapes, on this graph we can see how the Strain Energy exactly correlates to the Stiffness Sensitivity of the cantilever beam model, as it was shown in Figure 11, Chapter IV.

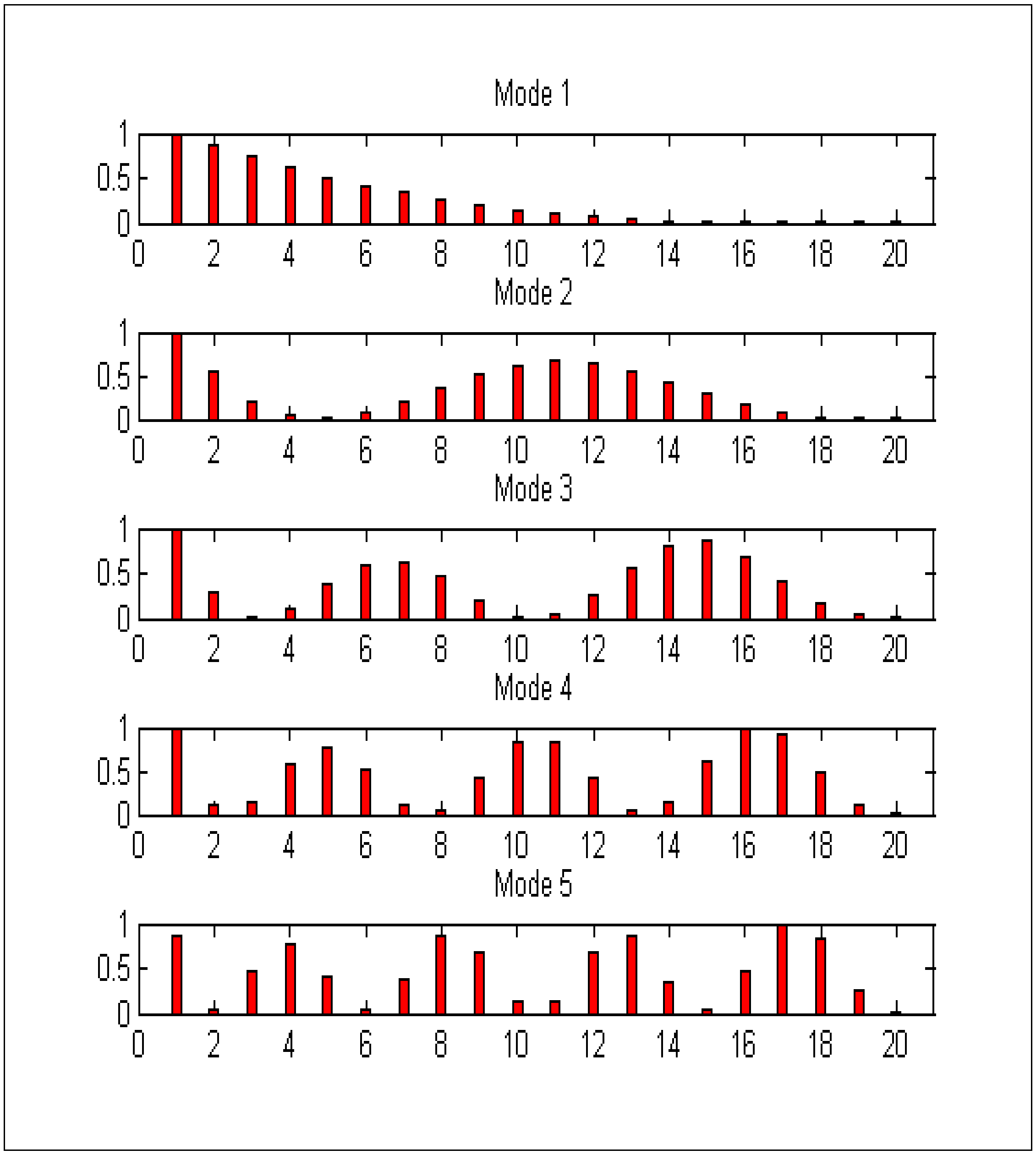

Figure 19. Strain Energy distribution base system.

The magenta color bars in Figure 20 depict the Kinetic Energy distribution along the twenty elements Base system of the beam over the first five mode shapes, on this graph we can conceivably see how the Kinetic Energy is exactly correlated to the Mass Sensitivity of the cantilever beam model, as it was shown in Figure 12, Chapter IV.

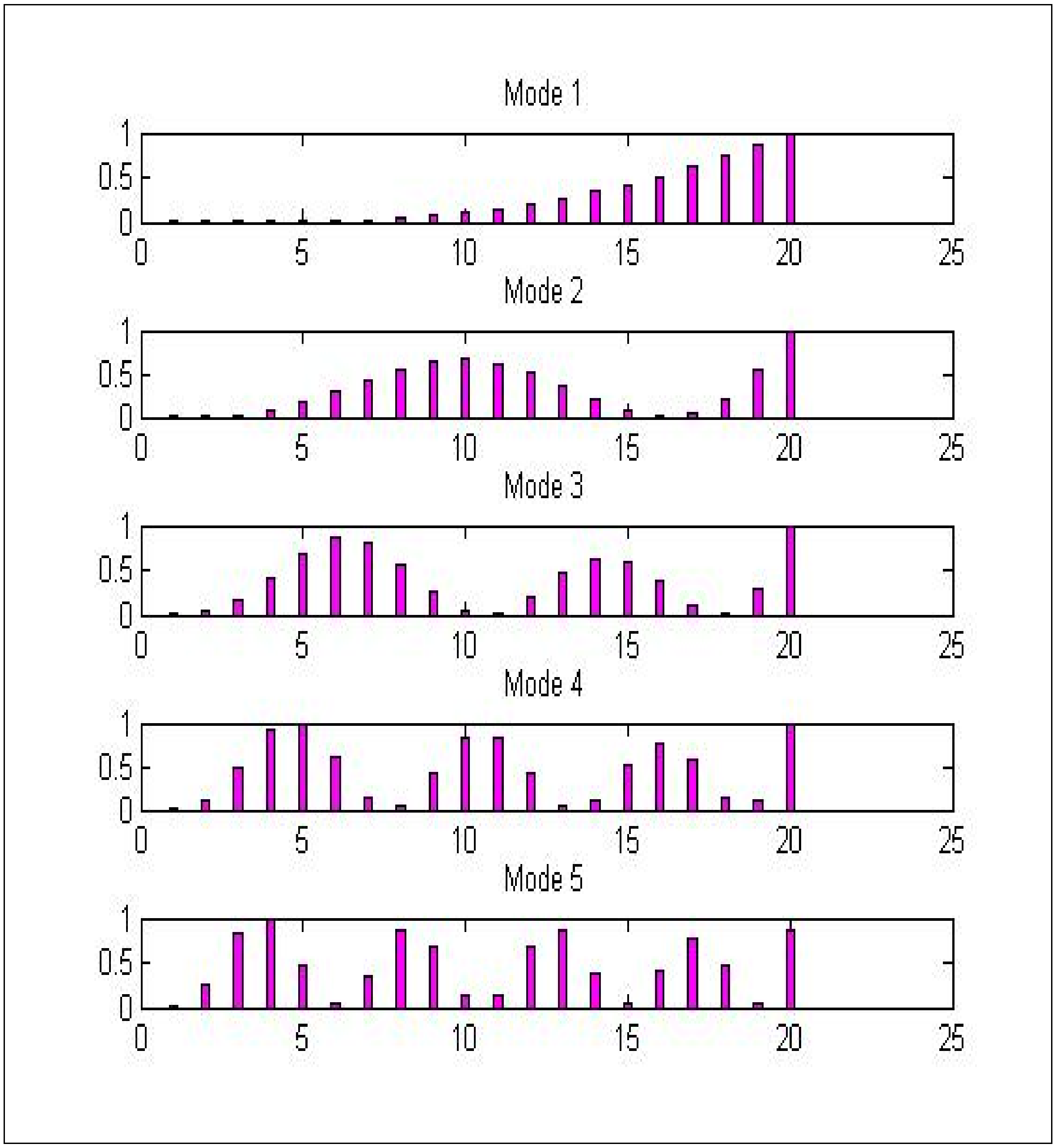

Figure 20. Kinetic energy distribution base system.

In the same way, a comparison of the Strain and Kinetic Energies to the Mass and Stiffness Sensitivities is performed with the ABC's applied.

Figure 21 depicts the Strain Energy distribution along the twenty elements with ABC applied at DOF 25 of the beam over the first five mode shapes, in this graph we can see how the Strain Energy distribution is exactly correlated to the Stiffness Sensitivity distribution of the cantilever beam model with the ABC applied at the same DOF, as it was showed in Figure 13, Chapter IV.

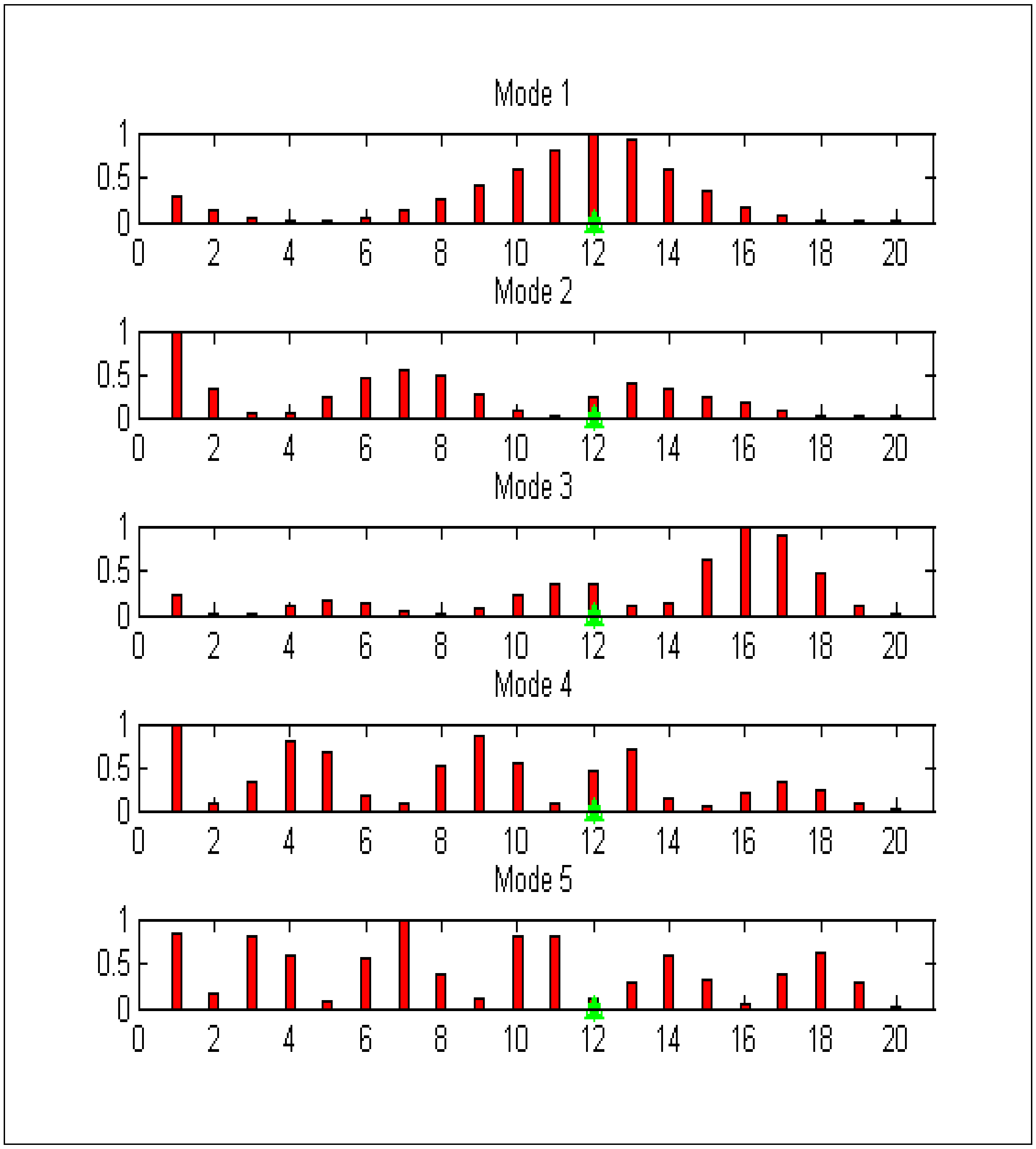

Figure 21. Strain energy distribution with ABC at DOF 25.

Figure 22 depicts the Kinetic Energy distribution along the twenty elements with ABC applied at DOF 25 of the beam over the first five mode shapes, on this graph we can see how the Kinetic Energy distribution is exactly correlated to the Mass Sensitivity of the cantilever beam model with the ABC applied at the same DOF, as it was showed in Figure 15, Chapter IV.

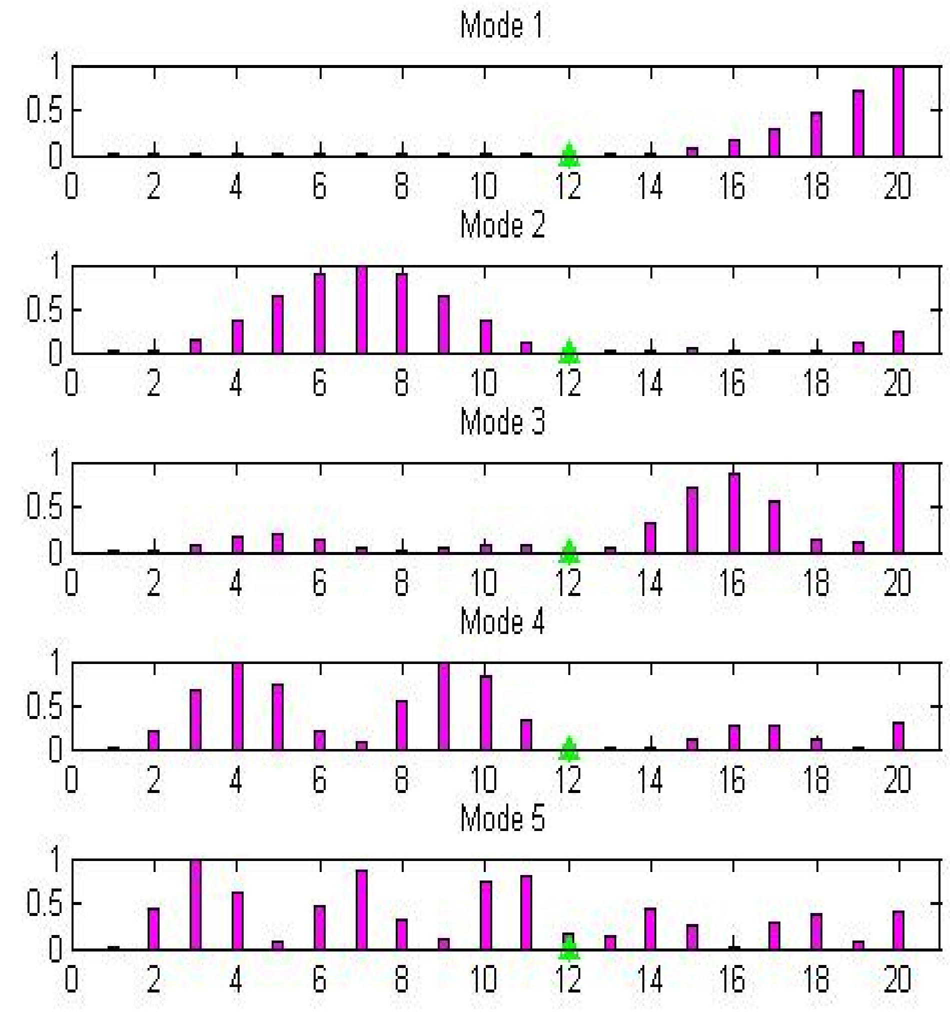

Figure 22. Kinetic energy distribution with ABC at DOF 25.

It has been proven for the cantilever beam modeled, that the Strain and Kinetic Energy matrices are directly correlated to the Stiffness and Mass Sensitivity matrices, respectively; therefore we can conclude that the same trend observed on the Stiffness Sensitivities distribution from the last chapter applies for the Strain Energy distribution, as well as the same trend observed for the Mass sensitivity distribution applies for the Kinetic Energy distribution.

It is also seen how the Stiffness Sensitivity matrix is directly correlated to the flexural properties of the structure and how the Mass Sensitivity matrix is directly correlated to the Kinetic energy of the structure, which in other words indicates it is a function of the velocity stored in the structure at each specific mode shape. That is why the Figures depicted of Mass Sensitivity and Kinetic Energy distributions showed low sensitivity at the hard points of the structure (i.e., high stiffness areas) and higher sensitivity on the portions away from the fixed points, where there is minimum Potential energy, but maximum Kinetic energy.

This is going to play an important role as it will be seen in the next chapter, as it is a very useful tool in the endeavor of predicting damage detection in the model.

VI. DAMAGE DETECTION

It was shown that there exists a direct correlation between the Mass sensitivity with respect to the Kinetic Energy matrices as well as the Stiffness sensitivity with regard to the Strain Energy matrices. Furthermore with the tendencies observed and discussed at the end of the last chapter when the ABC's were applied; several scenarios were run in order to find the best choice of ABC coordinates, such that it could result with the best error estimate of the structural model.

As the number of elements is augmented, the finite element model accuracy improves. The disadvantage is that as the number of elements increases, so does the computational time. The twenty element model was used to perform the error prediction calculation in the present thesis, making use of the first five modes to perform the analysis, as it was decided due to the reliable convergence, when the number of elements of the model gets increased, as it can be seen on Table 2.

# OF ELEMENTS	**10**	**20**	**42**
MODE 1	8.632305	8.632298	8.632298
MODE 2	54.09948	54.0978	54.09769
MODE 3	151.5137	151.4776	151.4752
MODE 4	297.1136	296.8493	296.8317
MODE 5	491.9194	490.7656	490.6868
MODE 6	736.9511	733.2692	733.0091
MODE 7	1033.978	1024.51	1023.809
MODE 8	1385.246	1364.733	1363.099
MODE 9	1791.106	1754.315	1750.902
MODE 10	2226.603	2193.797	2187.248

Table 2. Frequency (Hz) versus beam elements.

In order to run the experiments, two FEM beams were modeled in MATLAB, one being the analytical model and the other being the experimental, both models are exactly the same and correspond to the cantilever beam depicted in Figure 23, the beam specifications are those stated on Table 1.

The experimental beam was used to introduce a known perturbation at a known element as it is requested from the MATLAB computer program.

The errors implemented into the experimental beam consisted of a 10 percent of either mass or stiffness perturbations.

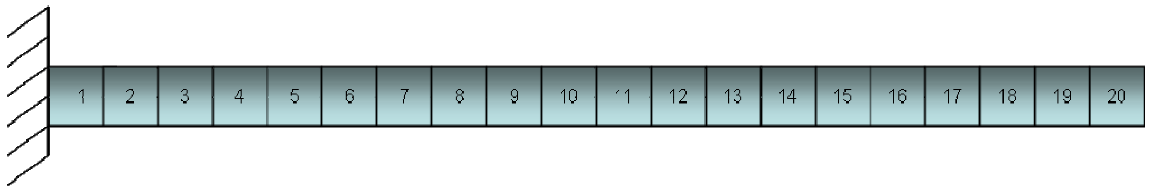

Figure 23. 20 Element Cantilever Beam.

A. EXPERIMENT

In order to find a set of ABC's, such that we could produce a set of modes with an improved sensitivity distribution, consequently a better error estimate; twenty four different cases were run to visualize the error perturbations introduced on the experimental cantilever beam model. Twelve scenarios consider the analysis of one ABC and one damaged element, and twelve scenarios consider the analysis of multiple ABC's and multiple damaged elements. Table 3 illustrates the experiment set up as follows:

ONE ABC/ ONE DAMAGED ELEMENT		**MULTIPLE ABC'S/ MULTIPLE DAMAGED ELEMENTS**	
TYPE OF DAMAGE	NUMBER OF CASES	TYPE OF DAMAGE	NUMBER OF CASES
STIFFNESS	6	STIFFNESS	6
MASS	6	MASS	6

Table 3. Twenty four Experiment Cases.

On the following cantilever beam drawings, the pinned ABC's are represented by the red triangles, the elements with the stiffness perturbations are represented in brown color and the elements with the corresponding mass perturbations are represented in magenta.

The selection of the ABC's was chosen following the sensitivity patterns identified and discussed on the last two chapters in relation to the mass and stiffness behaviors with respect to the Kinetic and Strain Energies performance, respectively.

The one ABC/ one damaged element cases are analyzed with 3 different graphs:

- The cantilever beam with the graphic error representation of the error, as well as the pinned DOF where the ABC is applied,
- The stiffness or mass sensitivity distribution along the beam for the specific case in analysis, and
- The actual damage detection (error prediction) of the model.

B. STIFFNESS DAMAGE DETECTION WITH ONE ABC

1. Case 1. Stiffness Error at Element 5, and ABC Pinned at DOF 9

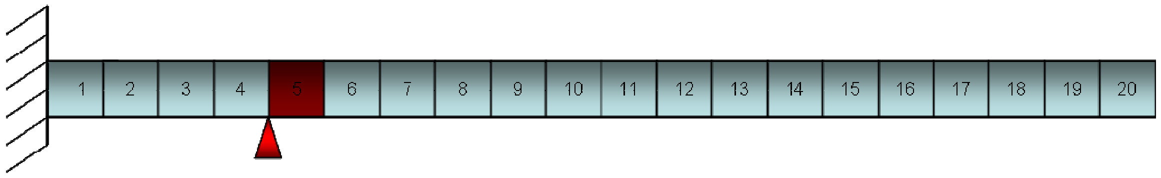

Figure 24. Stiffness error at element 5, ABC pinned at DOF 9.

Figure 25 shows the Stiffness sensitivity distribution along the beam represented by the red bars, the same figure also shows how the sensitivity is increased at the DOF's where the ABC is applied. Figure 26 represents the error localization at those places where the blue bars are raised. The stem plot with the red circle on the top indicates the magnitude and localization of the actual error.

Figure 26 indicates, how the damage detection is correctly identified, showing how areas of strong sensitivity are very prone to detect the stiffness errors imposed on the beam. Additionally the increased sensitivity due to the pinned ABC at a DOF adjacent to the element where the stiffness error perturbation is applied showed the effectiveness in the damage detection.

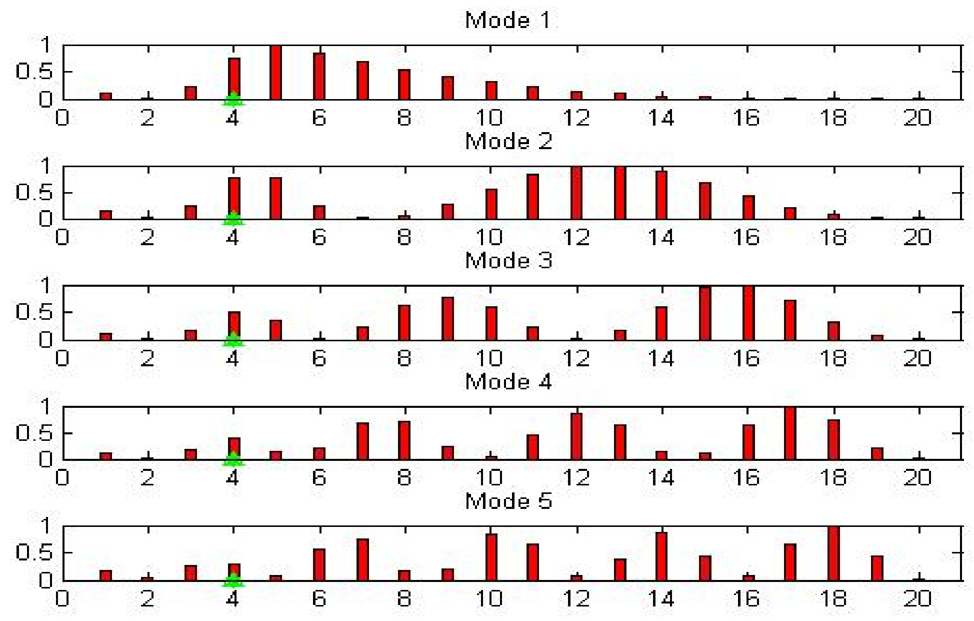

Figure 25. Stiffness sensitivity distribution Case 1.

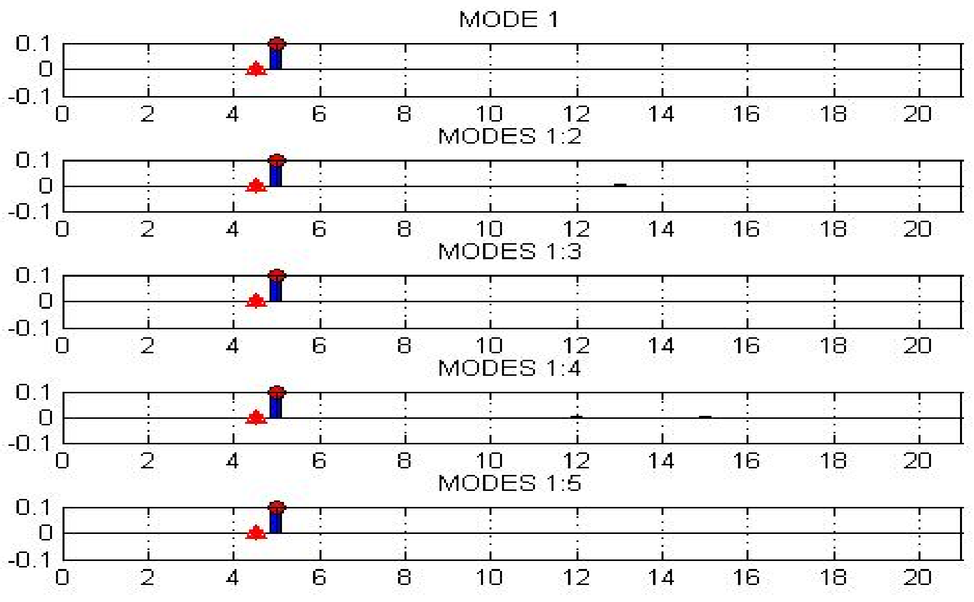

Figure 26. Stiffness error prediction Case 1.

2. Case 2. Stiffness Error at Element 5, and ABC Pinned at DOF 31

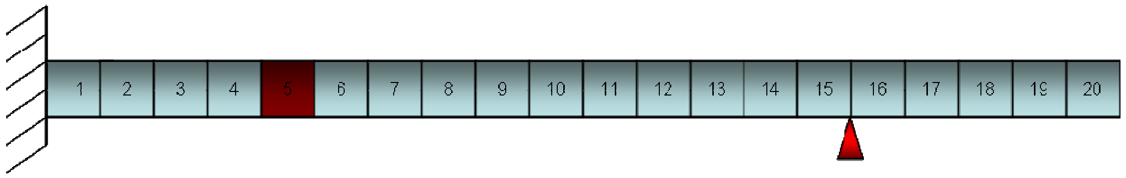

Figure 27. Stiffness error at element 5, ABC pinned at DOF 31.

Figure 28 shows the Stiffness sensitivity distribution along the beam represented by the red bars, this figure shows how the sensitivity is increased at the DOF where the ABC is imposed, which as for this case is located distant with respect to the element where the stiffness error is applied.

Even though the sensitivity bars in the damaged element region are low, the actual error is located. This situation is due to the fact that the Base sensitivity distribution (no ABC's applied) next to the fixed end of the beam is high, as it is shown in Figure 19, that is why the error prediction according to Figure 29 still preserves the damage when modes one through three up to modes one through five are retained.

This case shows how even when the pinned ABC is applied at a DOF further apart from the stiffness damaged element, where no increase in the sensitivity distribution is achieved, it still can be able to retain the error perturbation, as long as the Base sensitivity distribution is high enough at that particular section of the model.

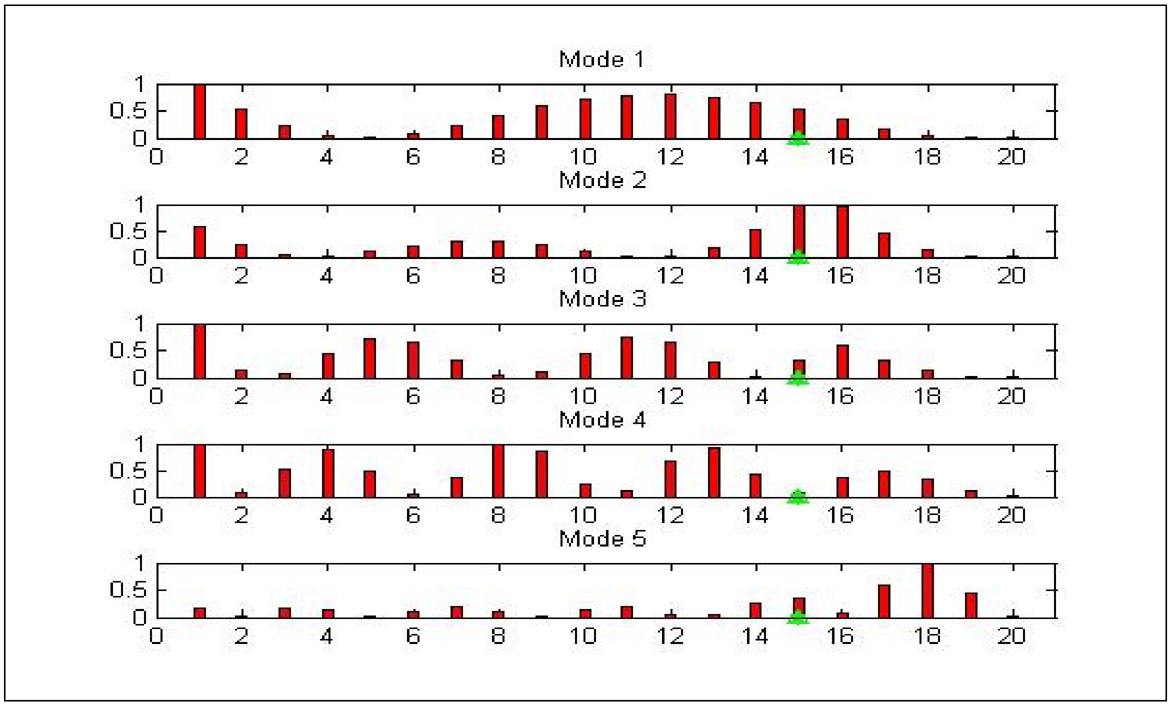

Figure 28. Stiffness sensitivity distribution Case 2.

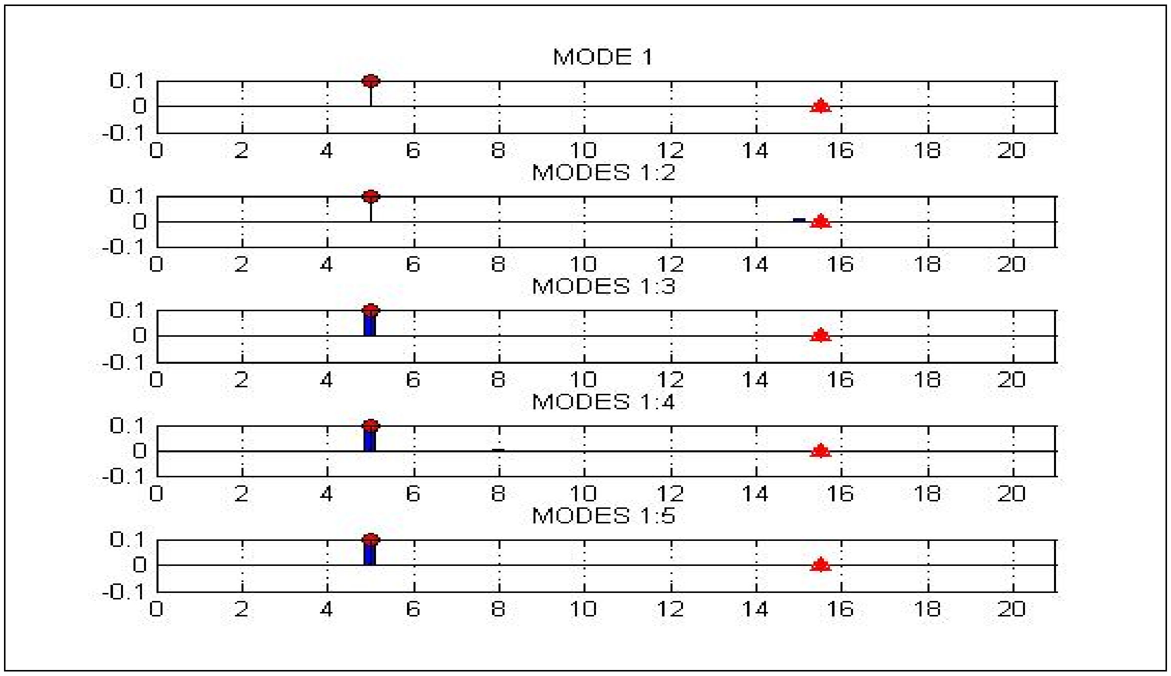

Figure 29. Stiffness error prediction Case 2.

3. Case 3. Stiffness Error at Element 10, and ABC Pinned at DOF 19

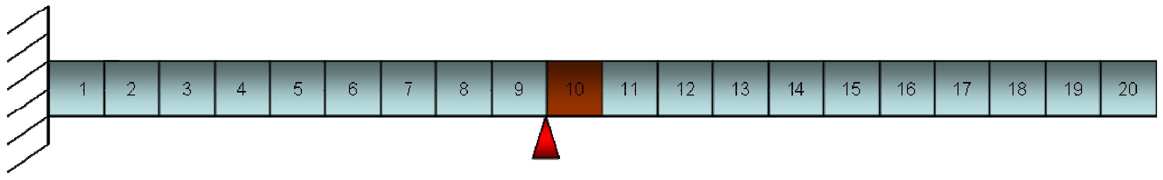

Figure 30. Stiffness error at element 10, ABC pinned at DOF 19.

Case 3 is exactly another representation of case 1, where it can be seen in Figure 31 the places of high sensitivity at the pinned DOF, which happens to coincide with the element among the actual stiffness error perturbation.

This situation again predicts the error in a consistent way although not as accurate as in case 1. When modes one through three were retained, the error was picked up accurately, retained modes one through four and one through five bounded the error at element nine, which is still close to the place of the actual error location.

One possible reason is due to the fact that areas of high strain energy concentration such as the fixed end of the cantilever beam are more likely to detect stiffness errors, thus the further away the damage element is from the fixed end of the cantilever beam model (area of high strain energy concentration), the more difficult the damage detection becomes. That is why the closest the damage element is to the fixed end, the better the damage detection will work due to the strain energy distribution. This is fortuitous, since the damage is more likely to develop near the fixed end, where the maximum stress concentration occurs (Panday, 1991).

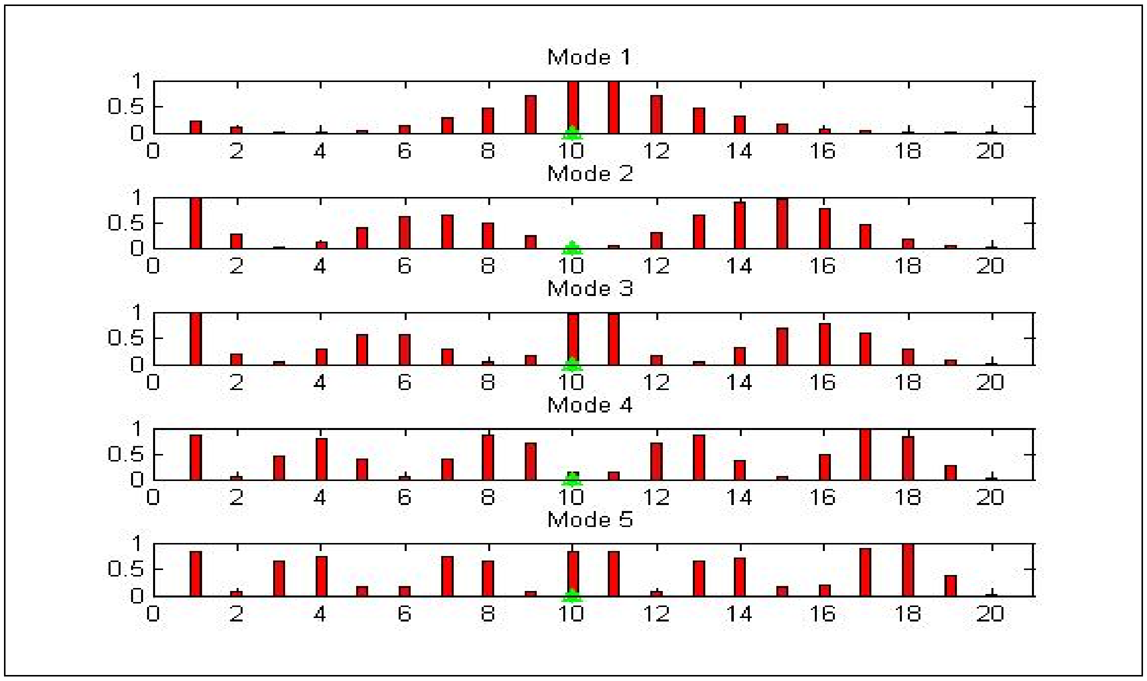

Figure 31. Stiffness sensitivity distribution Case 3.

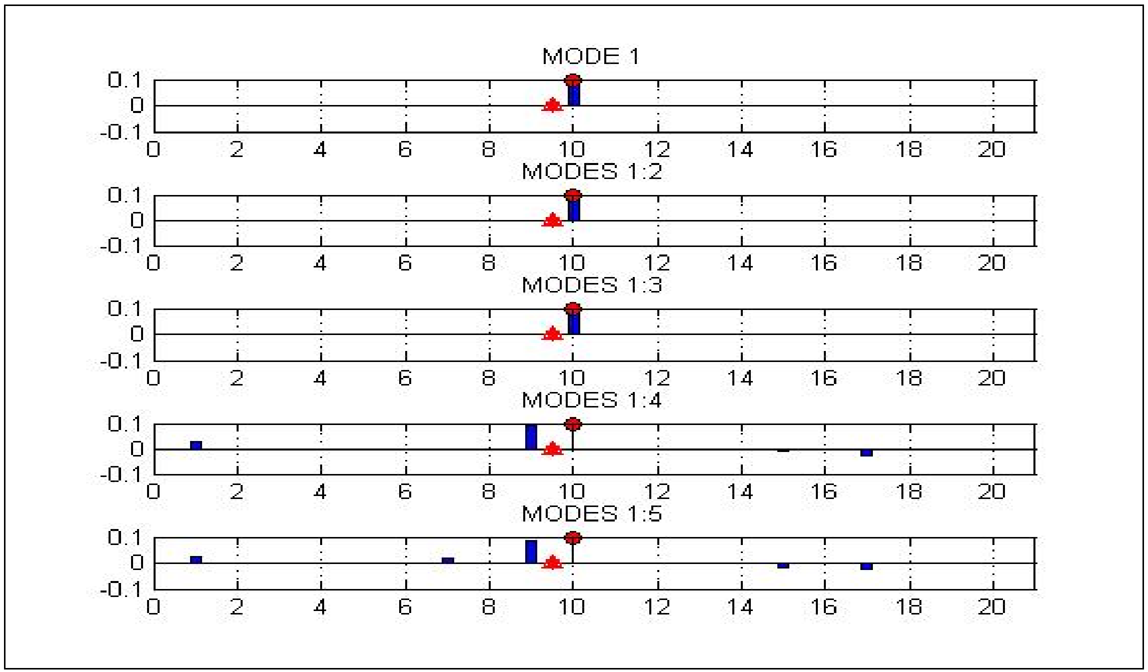

Figure 32. Stiffness error prediction Case 3.

4. Case 4. Stiffness Error at Element 10, and ABC Pinned at DOF 31

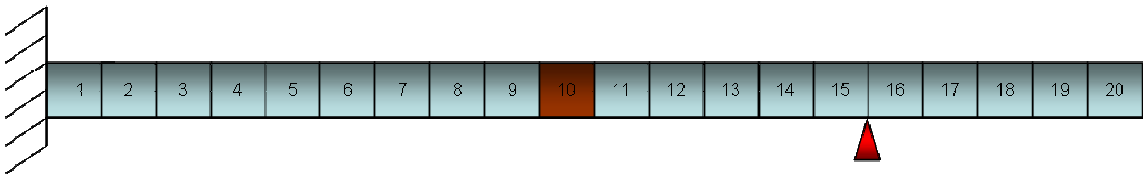

Figure 33. Stiffness error at element 10, ABC pinned at DOF 31.

Case 4 is another example related to case 2, where Figure 34, shows how the sensitivity is increased next to the DOF where the ABC is applied, which as for this case is located distantly with respect to the element where the stiffness error is exerted, although, this same figure also shows how there is still some sensitivity exposure along the damaged element.

The absence of blue bars in Figure 35 over the stem plot, shows how the error is never found, hence it confirms the notion that pinned ABC at a DOF further away from the element where the stiffness perturbation is applied is ineffective, in addition, since the Base Stiffness sensitivity or Base Strain Energy distributions are not really high at element 10, there are no possibilities that the error can be perceived.

This case clearly demonstrates how in distant areas (DOF's) where ABC is applied, the sensitivity is not very likely to be increased; hence no stiffness error is possible to be located.

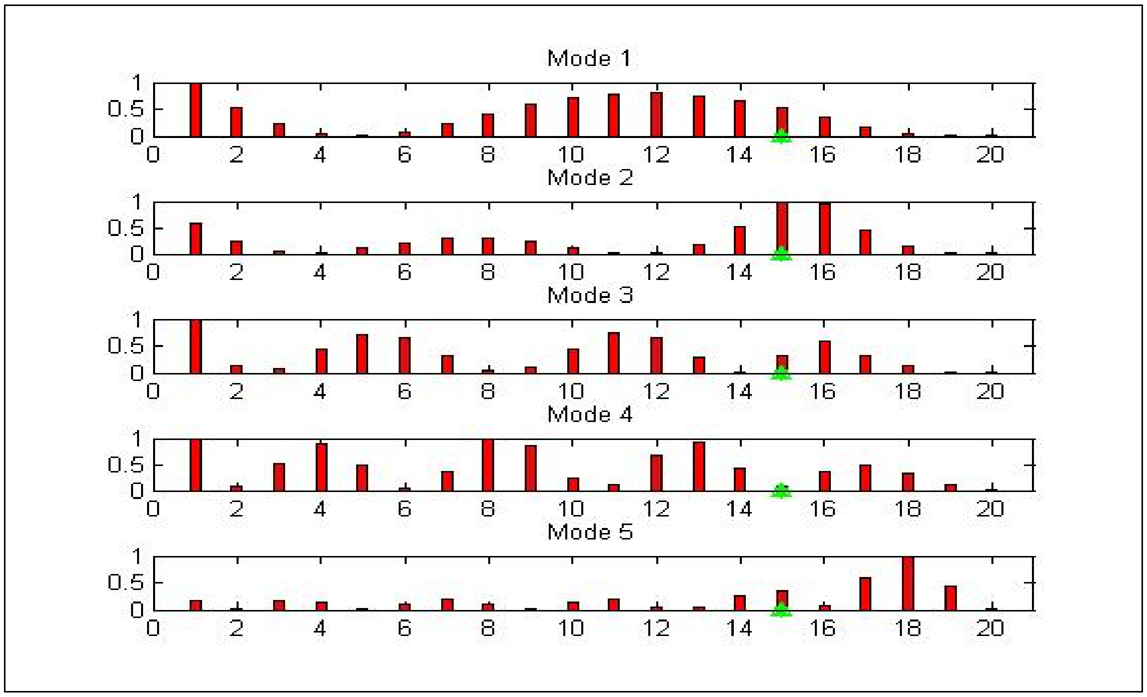

Figure 34. Stiffness sensitivity distribution Case 4.

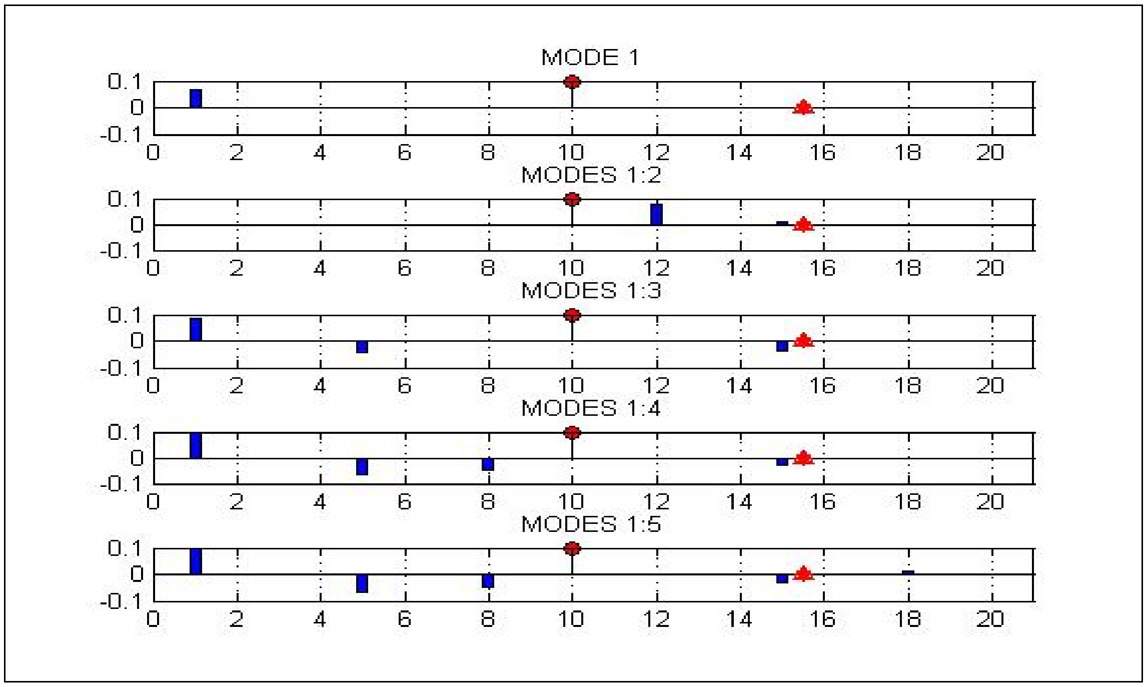

Figure 35. Stiffness error prediction Case 4.

5. Case 5. Stiffness Error at Element 15, and ABC Pinned at DOF 31

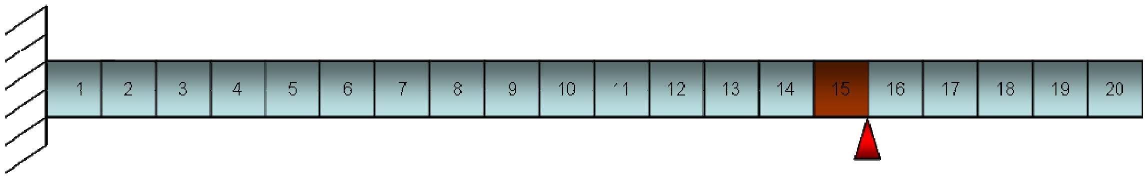

Figure 36. Stiffness error at element 15, ABC pinned at DOF 31.

Case 5 is analogous to cases 1 and 3 and concludes the set of representations of converged pinned ABC at the DOF location of the stiffness damaged element. This case again predicts the error in a consistent way, where the error detection is picked when the second mode is retained all the way up to retained mode five as it is depicted in Figure 38.

Cases 1, 3, and 5 prove the evidence found in Chapters IV and V where in particular for stiffness errors, when an ABC is particularly applied adjacent to the damaged element, the perturbation can be very well detected and retained, due to the increased sensitivity effect over the pinned ABC region.

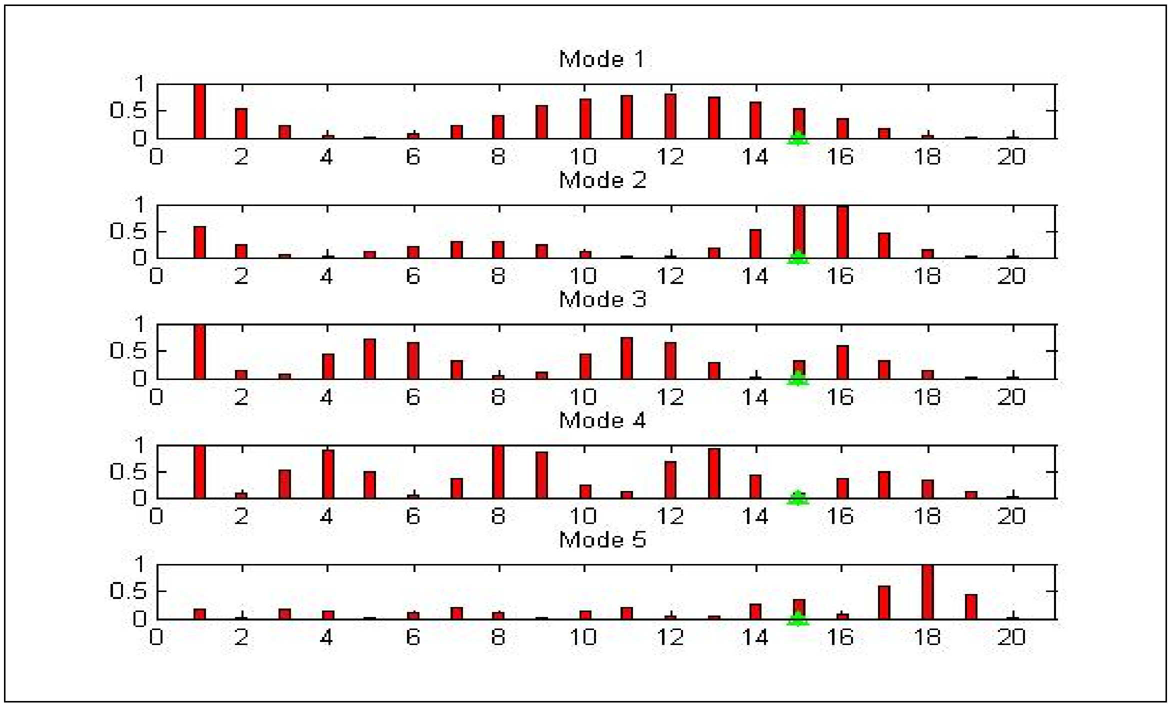

Figure 37. Stiffness sensitivity distribution Case 5.

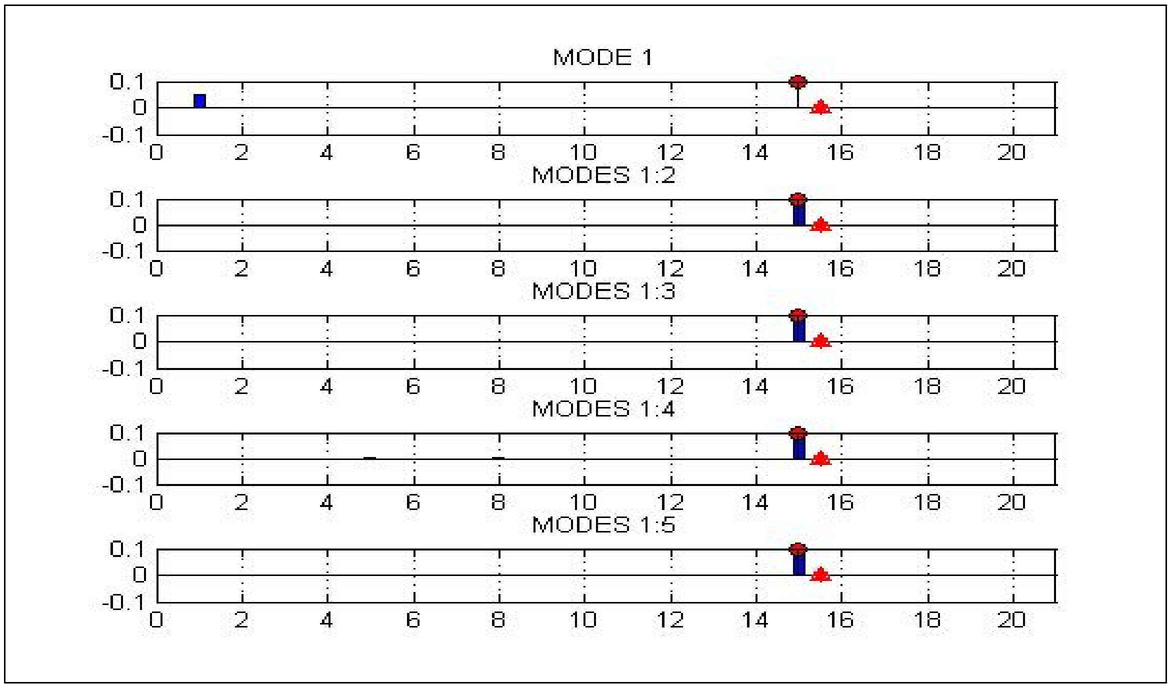

Figure 38. Stiffness error prediction Case 5.

6. Case 6. Stiffness Error at Element 15, and ABC Pinned at DOF 11

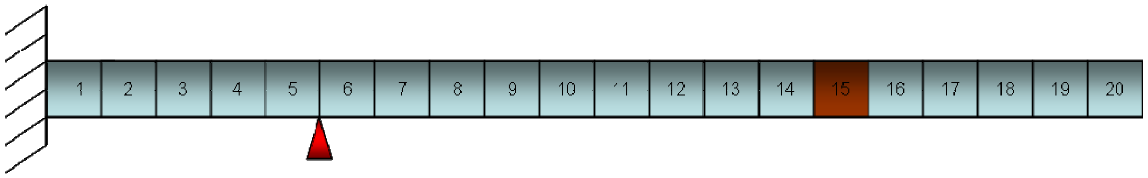

Figure 39. Stiffness error at element 15, ABC pinned at DOF 11.

Case 6 is the preceding and last description of cases 2 and 4, where the pinned ABC is applied at a DOF further away from the location of the stiffness damaged element.

Figure 40 shows how the sensitivity varies arbitrarily with respect to the damage element, hence is very unlikely to have a reliable prediction of the error, as it is the case shown in Figure 41, nevertheless the error happened to get retained at modes one to four and modes one to five. Even though this was the situation aroused for this specific case, there is no consistent statement that this pinned ABC set up is going to accurately predict the stiffness error on the cantilever beam model.

In summary, Cases 2, 4 and 6 ratify that when ABC's are applied at a DOF further away from the element in error, there is not advantage of the increased stiffness of the fixed condition effect, therefore the sensitivity is not considerably raised; thus no stiffness error is likely to be found.

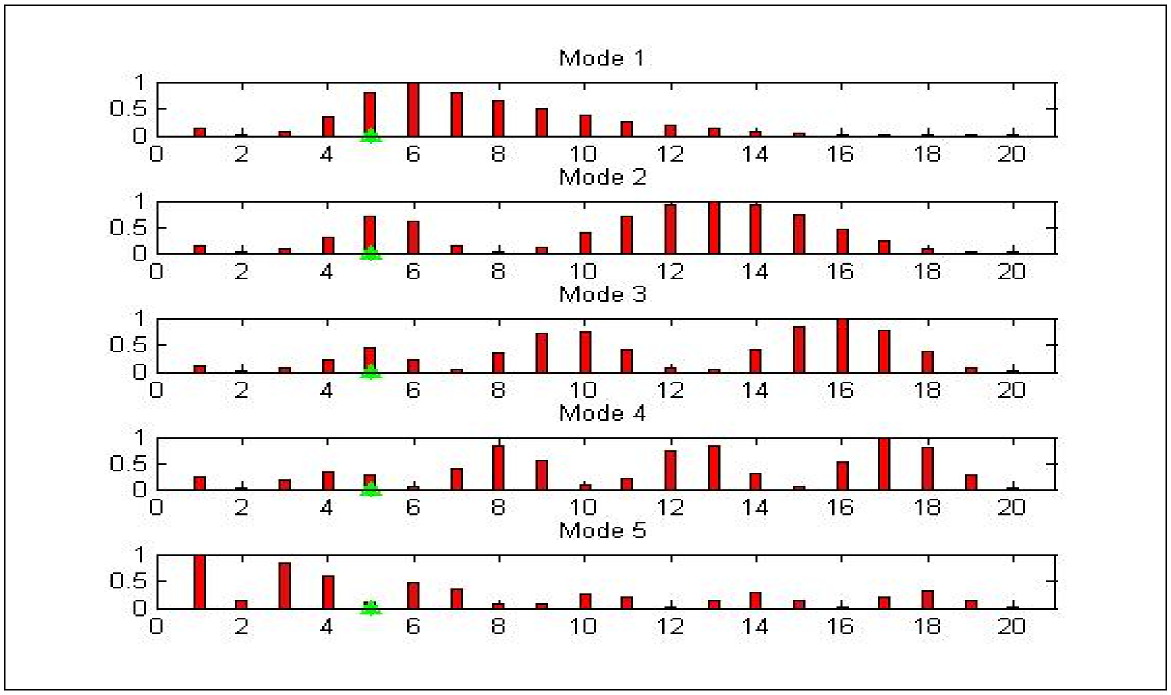

Figure 40. Stiffness sensitivity distribution Case 6.

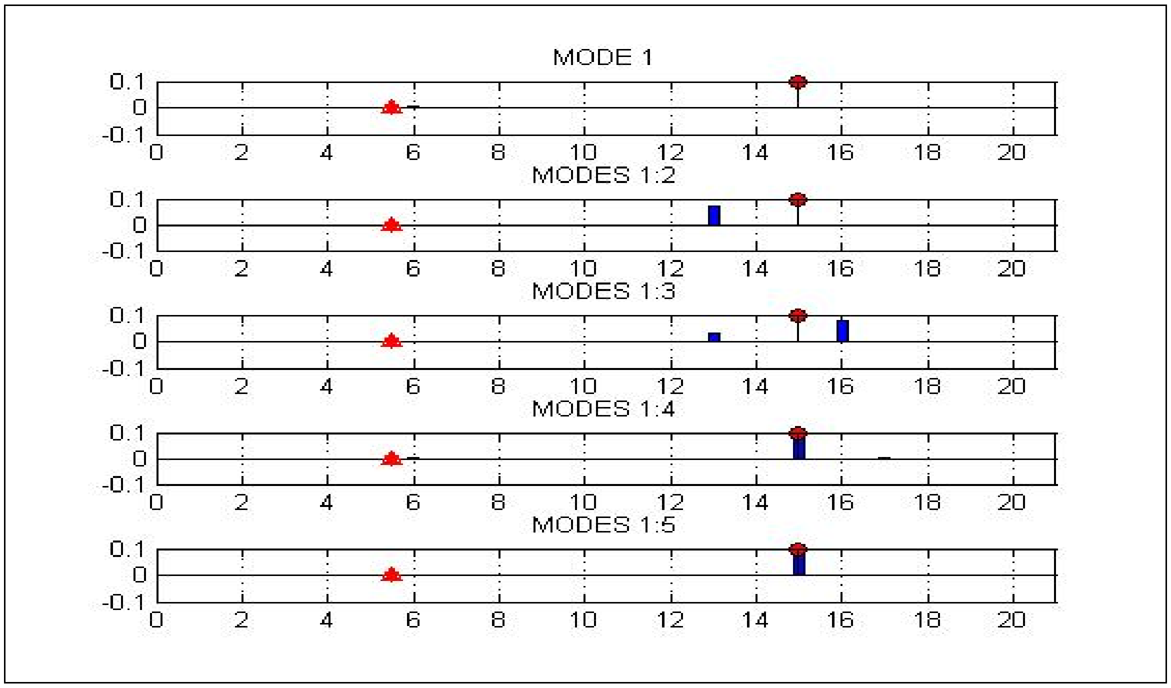

Figure 41. Stiffness error prediction Case 6.

C. MASS DAMAGE DETECTION WITH ONE ABC

1. Case 1. Mass Error at Element 5, and ABC Pinned at DOF 11

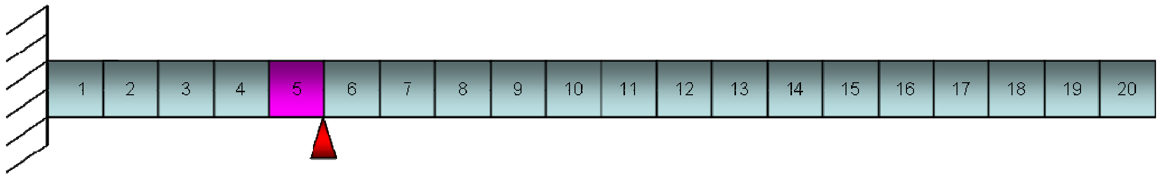

Figure 42. Mass error at element 5, ABC pinned at DOF 11.

Figure 43 shows the Mass sensitivity distribution along the beam represented by the magenta bars, this figure also shows how the sensitivity is decreased at the DOF where the ABC is applied. Figure 44 represents the error localization at those places where the blue bars are raised. The stem plot with the green circle on the top indicates the magnitude and localization of the actual mass error.

As for this case, Figure 44 indicates, how no damage detection is identified, since no blue bars are visible along the twenty element beam model, showing how areas of low sensitivity are not going to detect the mass errors imposed on the beam.

Since the mass perturbation error is directly correlated to the Kinetic Energy distribution of the beam, the mass error is a function of the velocity (inertia) of the model. Consequently the possibility of detecting mass errors at DOF's where the ABC coincides with the element with the mass error perturbation are minimal. This artificial manipulation implies that the corresponding DOF is pinned to ground, therefore no displacement is very likely to occur.

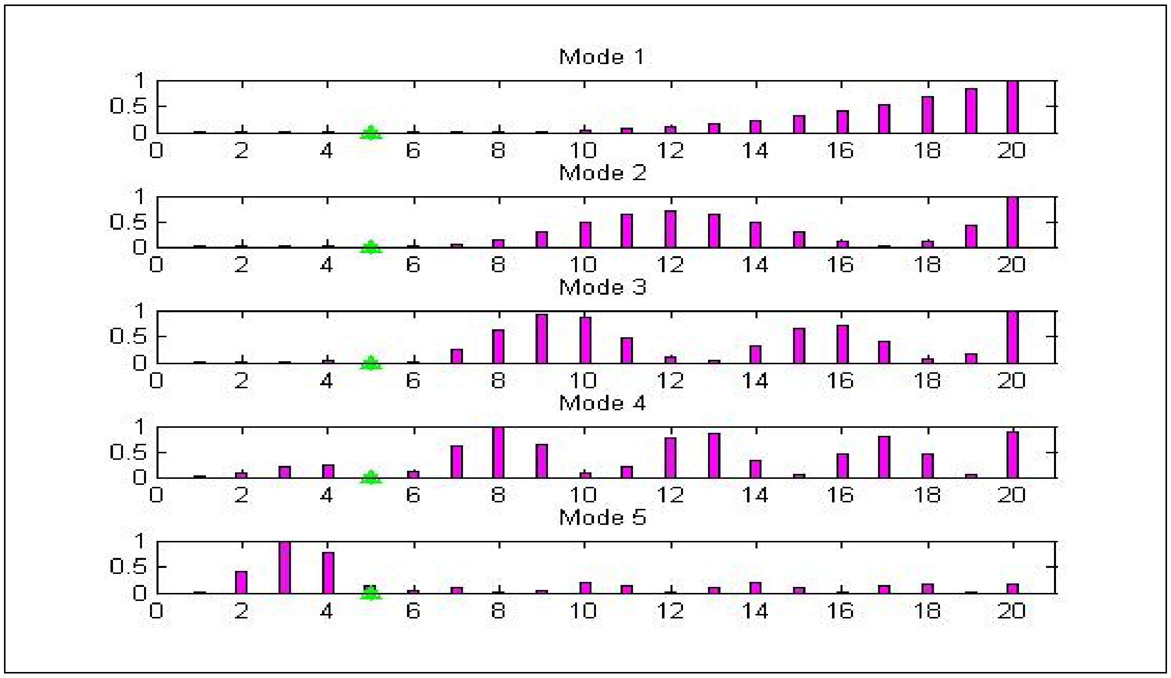

Figure 43. Mass sensitivity distribution Case 1.

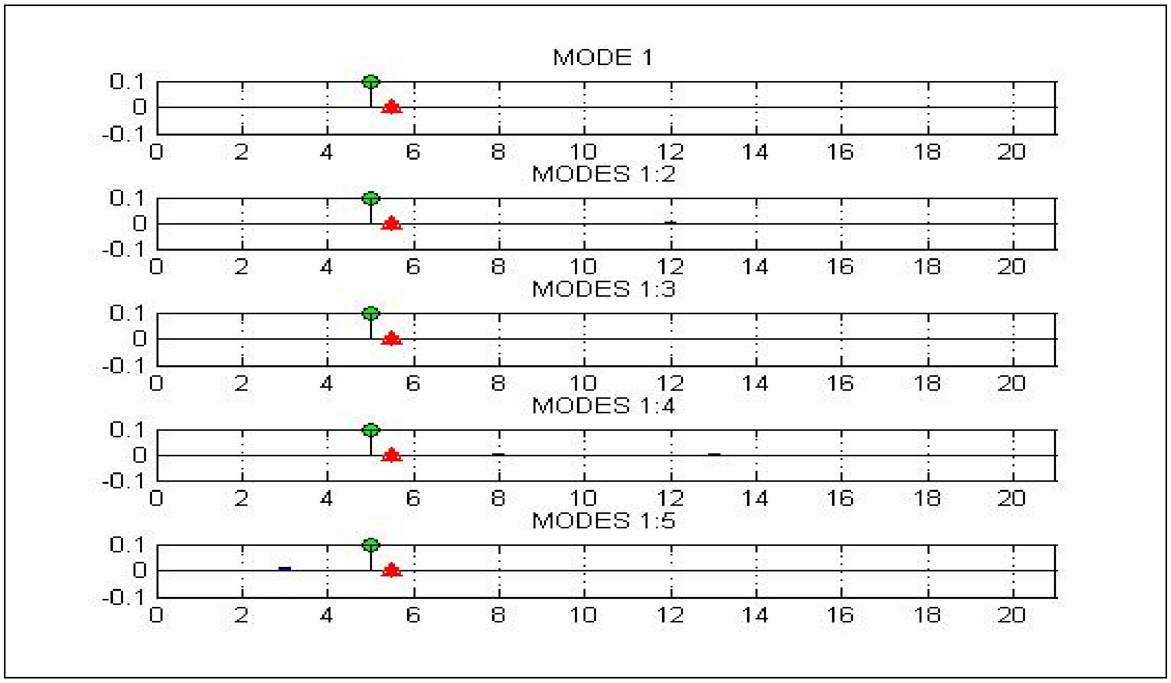

Figure 44. Mass error prediction Case 1.

2. Case 2. Mass Error at Element 5, and ABC Pinned at DOF 19

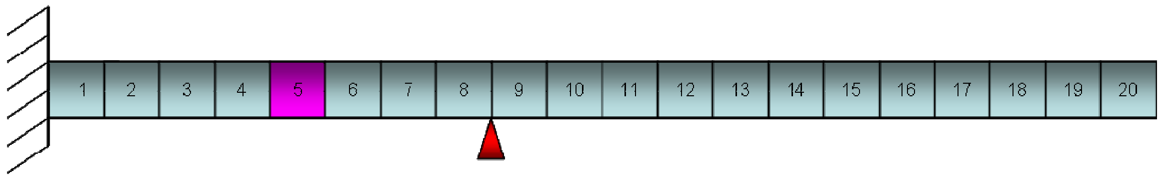

Figure 45. Mass error at element 5, ABC pinned at DOF 19.

Figure 46 shows the mass sensitivity distribution along the beam represented by the magenta bars, in the same figure it is also seen how the sensitivity is pushed away from the DOF where the ABC is applied, and how the sensitivity is increased at those DOF where the maximum displacement is present.

Since element five is comprised within the area of maximum displacement, therefore good sensitivity is present at those DOF. The blue bars on Figure 47 depict how the error is well retained in magnitude and location when the first three modes are retained. This scenario shows how the mass perturbation errors are a function of the maximum displacement along the beam.

Given that the mass perturbation error is correlated to the Kinetic energy distribution along the beam, the areas where displacement is evident, will also represent areas of Kinetic Energy presence; therefore the error is very likely to be predicted.

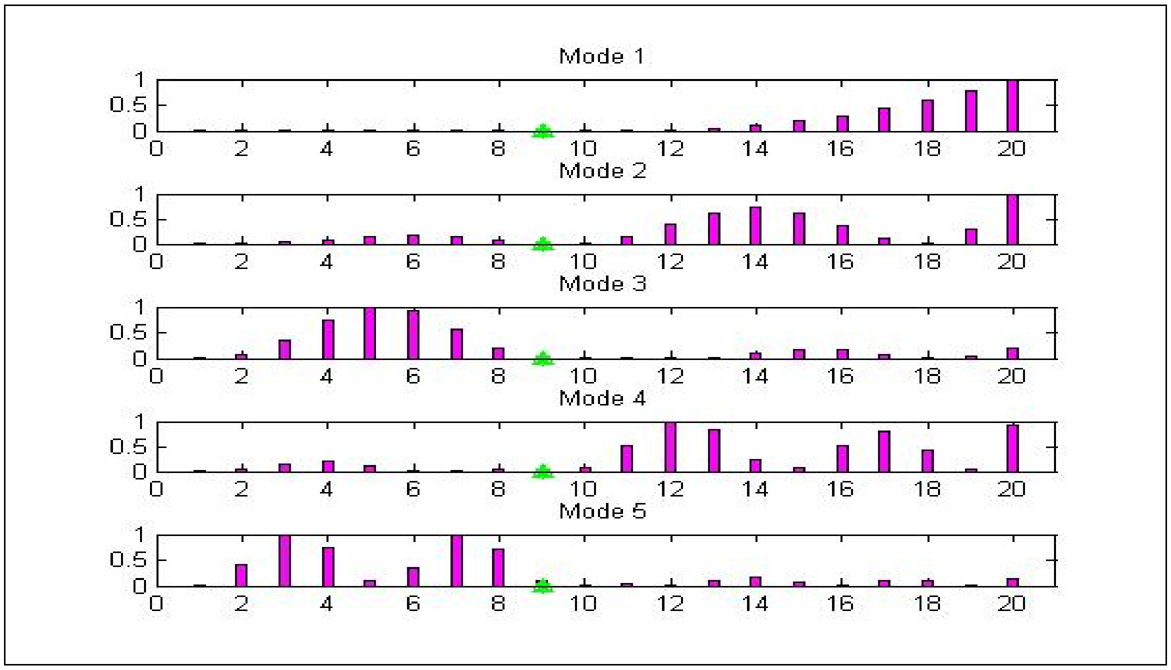

Figure 46. Mass sensitivity distribution Case 2.

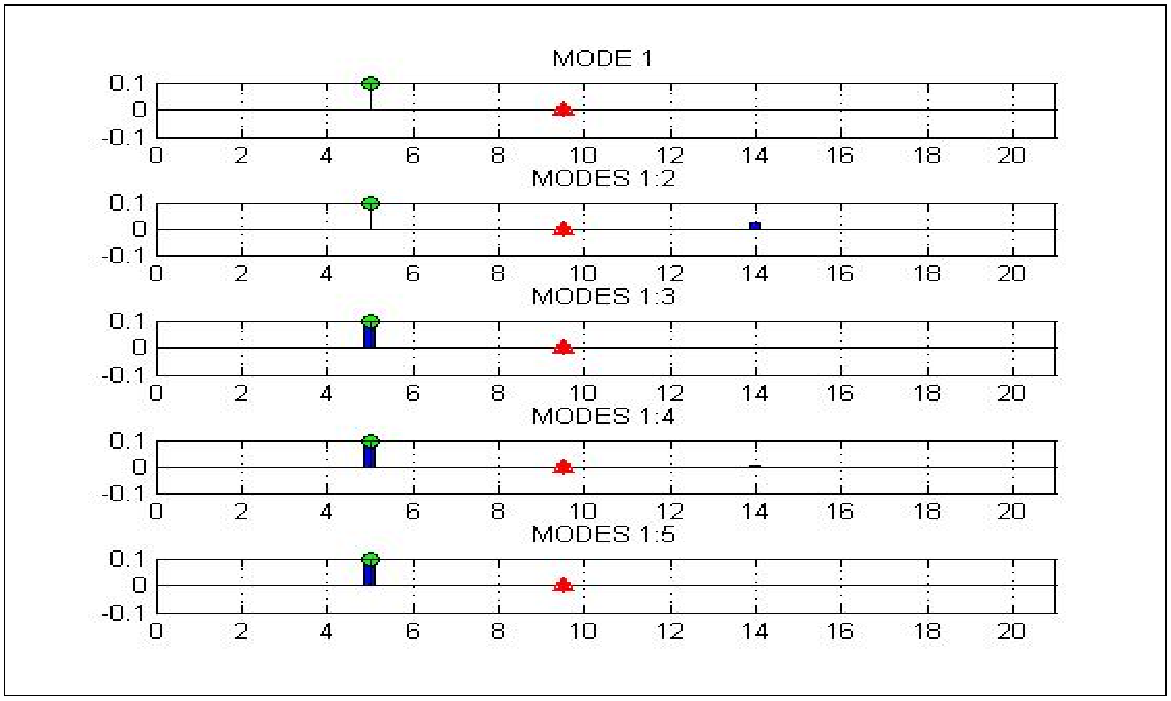

Figure 47. Mass error prediction Case 2.

3. Case 3. Mass Error at Element 10, and ABC Pinned at DOF 19

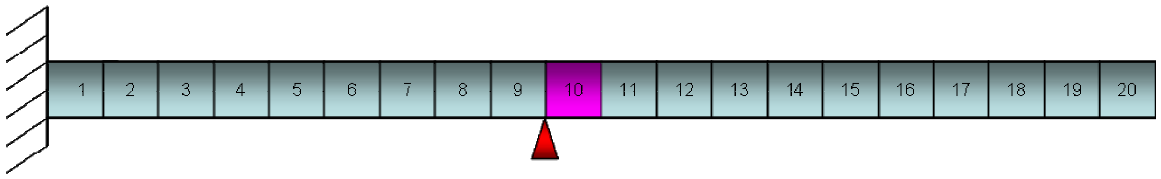

Figure 48. Mass error at element 10, ABC pinned at DOF 19.

Case 3 is analogous to case 1, where the pinned DOF happens to overlap with the element with the actual mass error perturbation. Figure 49 clearly shows how the sensitivity is diminished at the DOF where the ABC is applied.

The absence of blue bars on Figure 50 evidently proves again how no damage detection is identified. This case once again shows how areas of low sensitivity are not going to detect the mass errors imposed on the beam. Again, the lack of displacement effect due to the ABC pinned at a DOF adjacent to the element mass perturbation demonstrates how the mass error prediction is not likely to be successful.

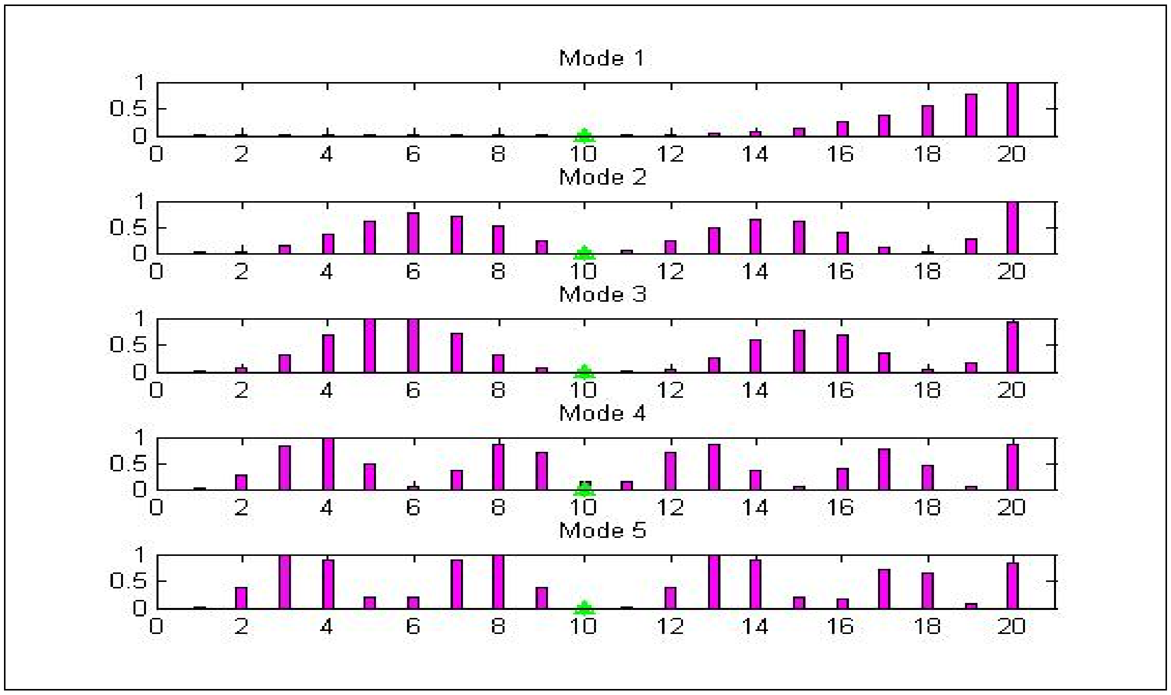

Figure 49. Mass sensitivity distribution Case 3.

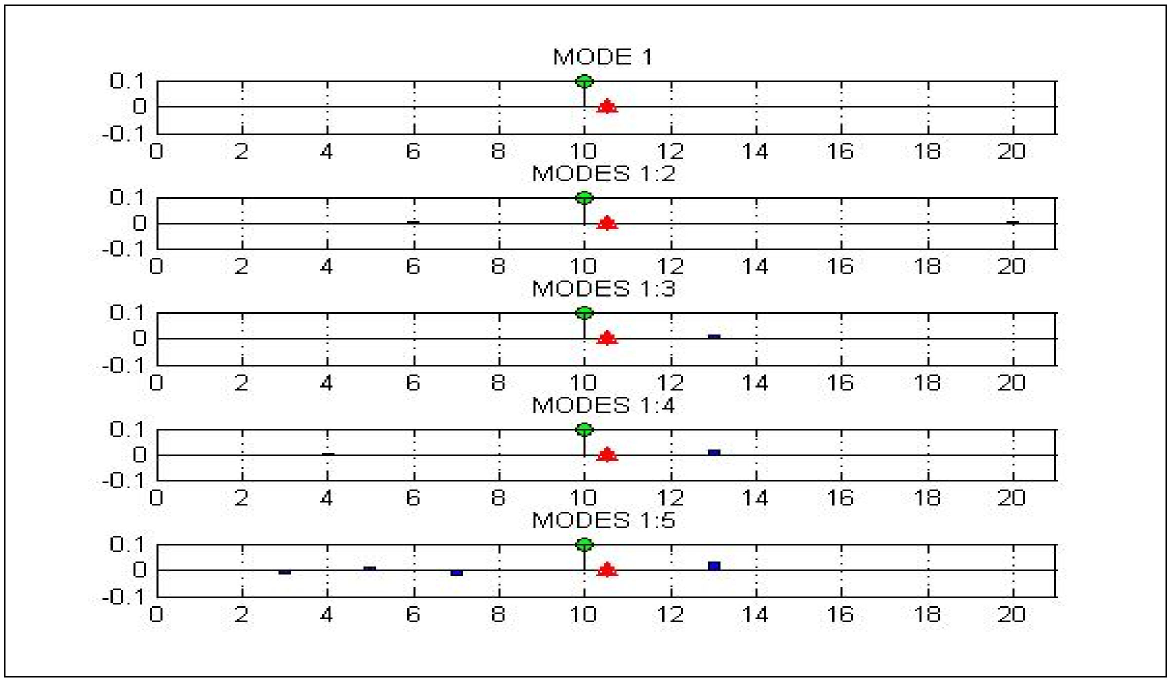

Figure 50. Mass error Prediction Case 3.

4. Case 4. Mass Error at Element 10, and ABC Pinned at DOF 31

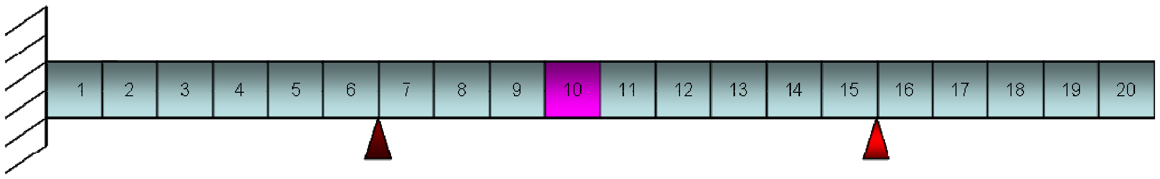

Figure 51. Mass error at element 10, ABC pinned at DOF 31 and DOF 13.

Case 4 happened to be a very sturdy scenario on finding the mass damage detection. It was assumed that the pinned ABC that would generate one of the highest mass sensitivities measurements was that at DOF 31. Figure 52 shows the mass sensitivity distribution along the beam model for the mentioned pinned DOF. But Figure 53 showed how the damage detection was never found, hence forcing to run the scenario for a variety of pinned ABC conditions.

After several runs pinned ABC at either DOF 13 or DOF 15 (dark red triangle in Figure 51), proved to predict the mass error perturbation exactly in the same approach as it is depicted on Figure 55, where the magnitude and location of the error is found when the first three modes are retained and hold the error up to the five retained modes.

DOF 13 happens to stay literally to the same distance with respect to element 10, as DOF 31 is, but the error prediction pattern was totally different when analyzed, where DOF 13 proved to do an acceptable error prediction, but the pinned ABC at DOF 31 did not work as efficiently as was expected.

Although the evidence found in Chapters IV and V still holds true, that is, areas of higher displacements are very likely to predict better the mass errors, it is cumbersome, how it will not work for every case where the higher displacements are evident, making this situation a problem of major concern where mass error detection is of interest.

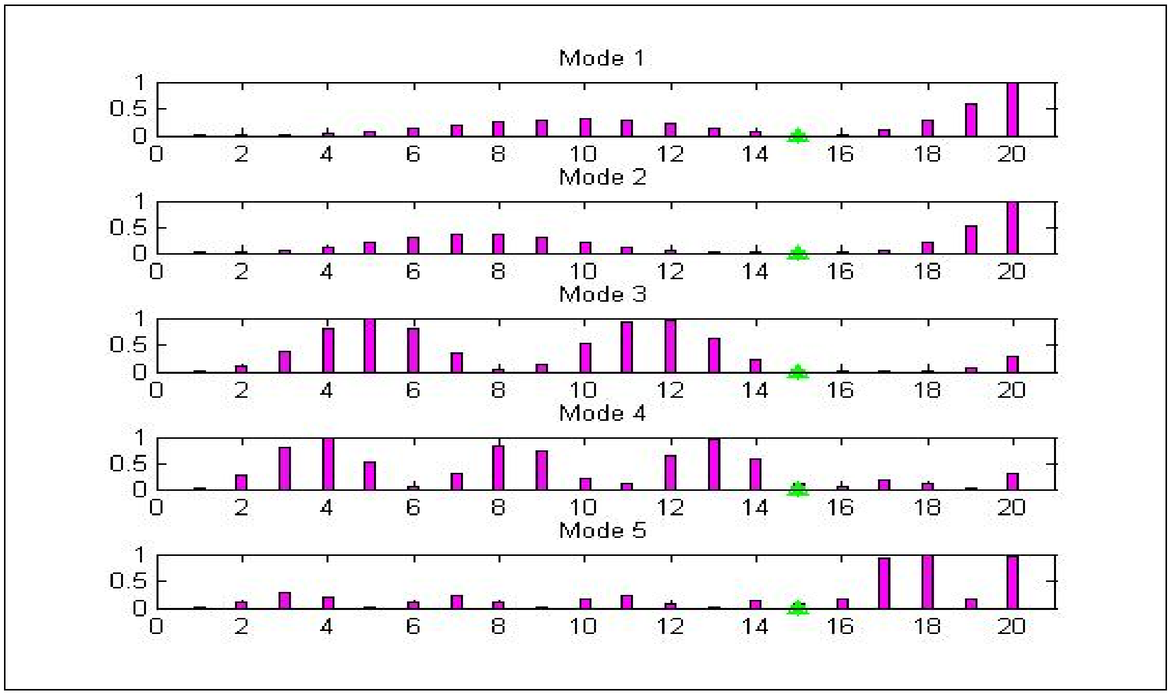

Figure 52. Mass sensitivity distribution Case 4 at DOF 31.

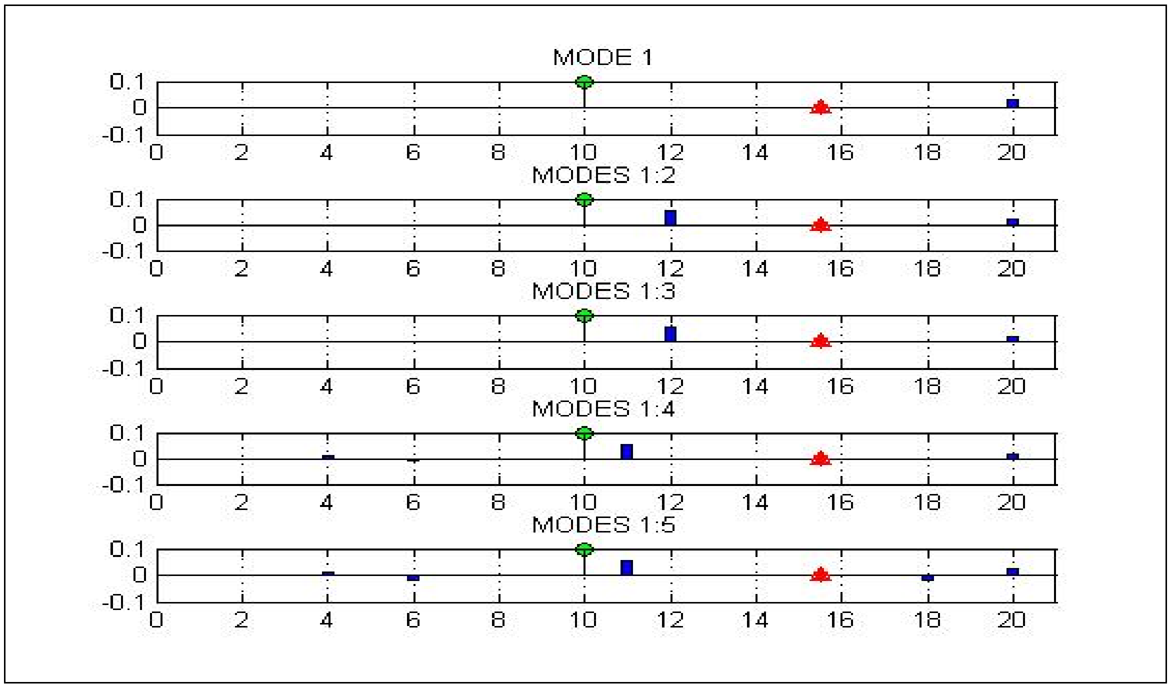

Figure 53. Mass error prediction case 4 at DOF 31.

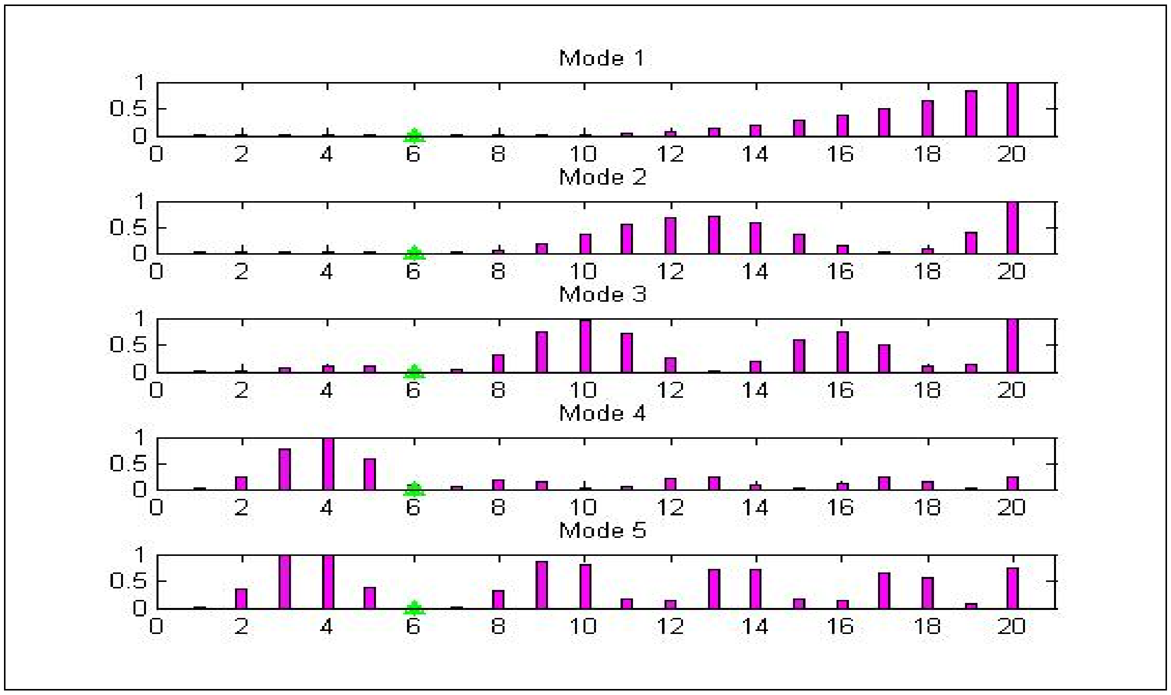

Figure 54. Mass sensitivity Case 4 at DOF 13.

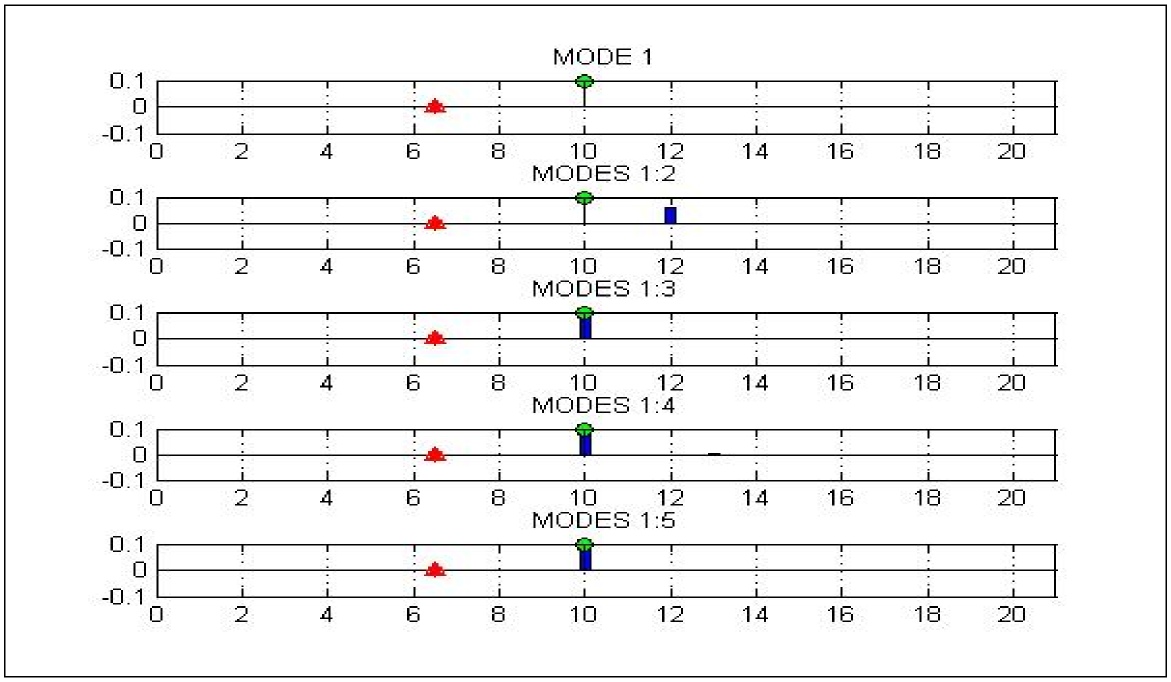

Figure 55. Mass error prediction case 4 at DOF 13.

5. Case 5. Mass Error at Element 15, and ABC Pinned at DOF 31

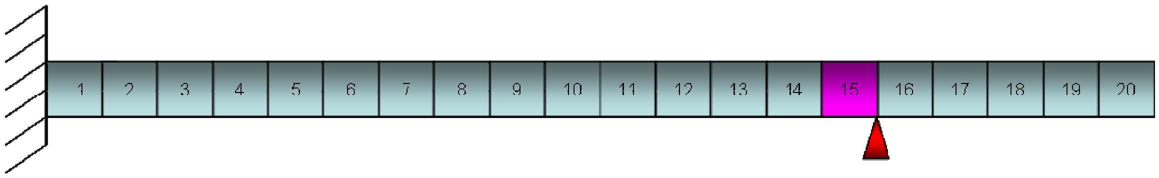

Figure 56. Mass error at element 15, ABC pinned at DOF 31.

Case 5 is the last representation of cases 1 and 3, where the pinned DOF lies adjacent to the element with the actual mass error perturbation. Figure 49 clearly shows how the sensitivity is diminished at the DOF where the ABC is applied.

Figure 50 proves again how no damage detection is identified. This case consolidates the reflection of how areas of low sensitivity are not going to detect the mass errors imposed on the beam. Again, the lack of displacement effect due to the ABC pinned at a DOF adjacent to the element mass perturbation demonstrates how the mass error prediction is never going to be successful.

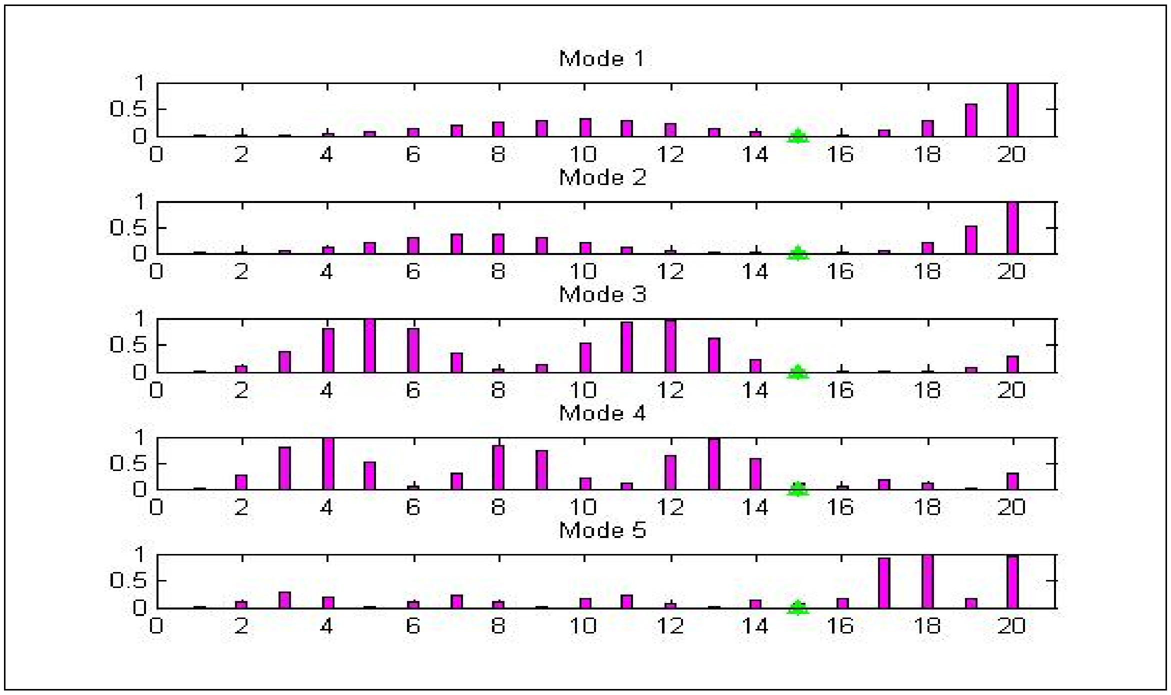

Figure 57. Mass sensitivity distribution Case 5.

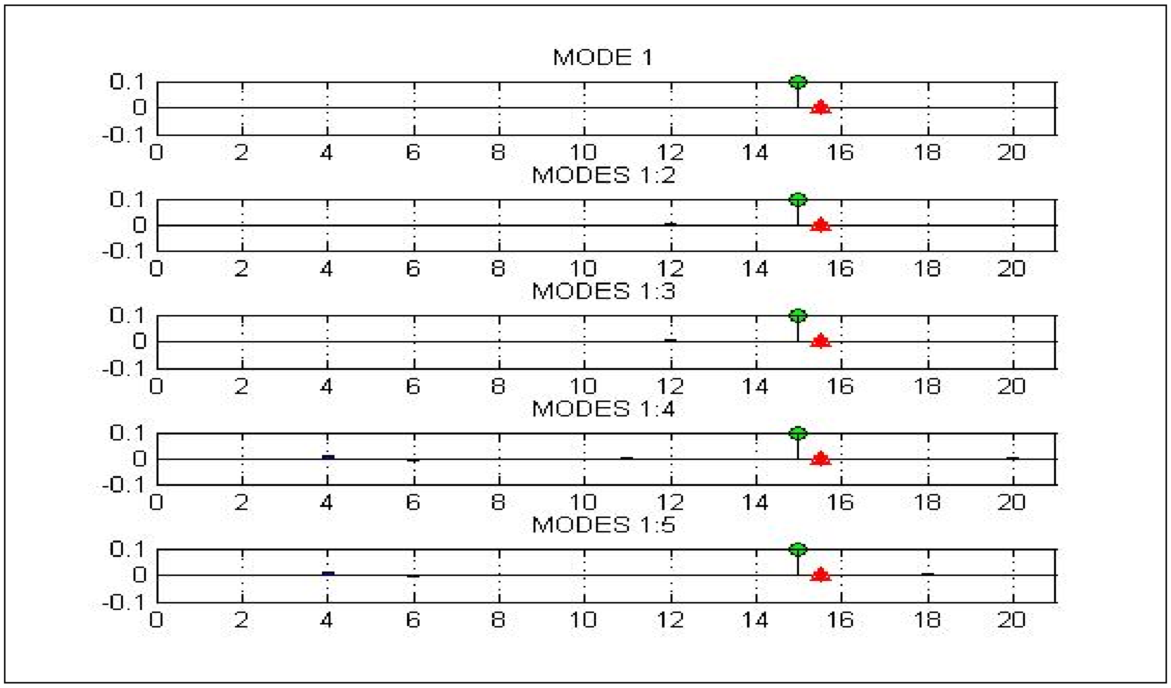

Figure 58. Mass error prediction Case 5.

6. Case 6. Mass Error at Element 15, and ABC Pinned at DOF 23

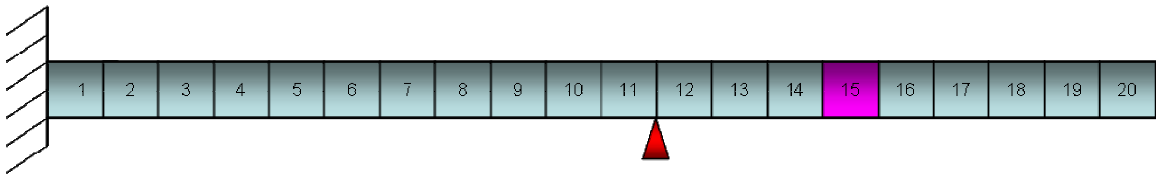

Figure 59. Mass error at element 15, ABC pinned at DOF 23.

Case 6 represented a very similar situation to the one experienced in case 4. Initially DOF 19 was selected as the pinned ABC condition to analyze, as it was assumed to generate a good Mass sensitivity distribution, but when the experiment was evaluated, it did not find the error accurately. Hence as in case 4, this case was run for a variety of pinned ABC's conditions.

The analysis was organized in two parts. The first with the ABC pinned to the left hand side of the mass damaged element, giving the boundary condition effect of a cantilever beam model. After several runs pinned ABC at DOF 23, was the best ABC configuration able to find the mass perturbation error, as it is shown on Figure 61, where modes one to three and one to four, retained the error in magnitude and location, mode five, lost the actual error detection, but still found some mass perturbation at elements fourteen and sixteen of the experimental cantilever beam.

The second part of the analysis was with the ABC set up to the right hand side of the mass damaged element, giving the boundary condition effect of a clamped- clamped beam. The best error prediction was found at pinned ABC at DOF 35 and 37, which found the error at mode five only, the rest of the DOF did not find the error at all, and no error prediction pattern proved to exist.

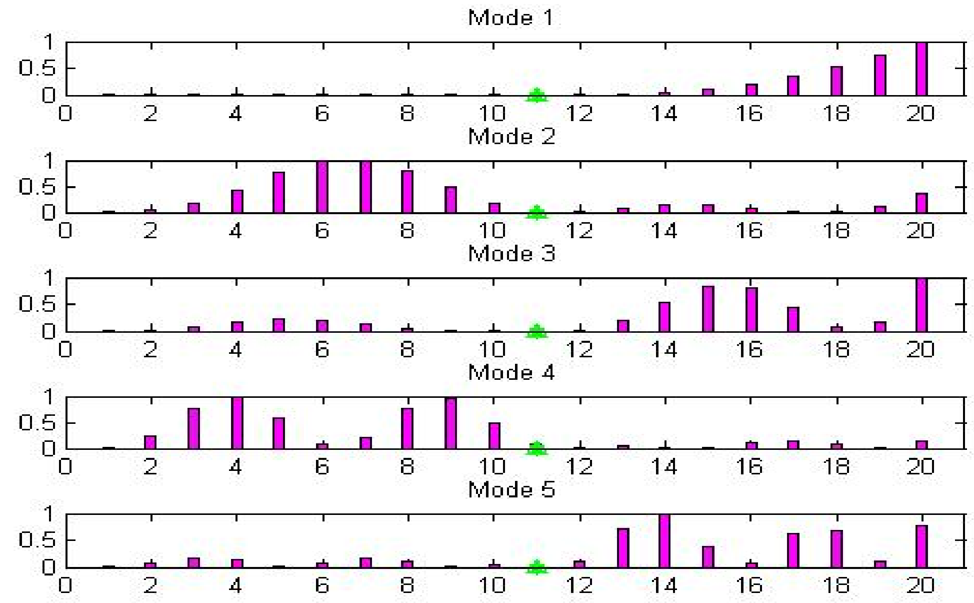

Figure 60. Mass sensitivity distribution Case 6.

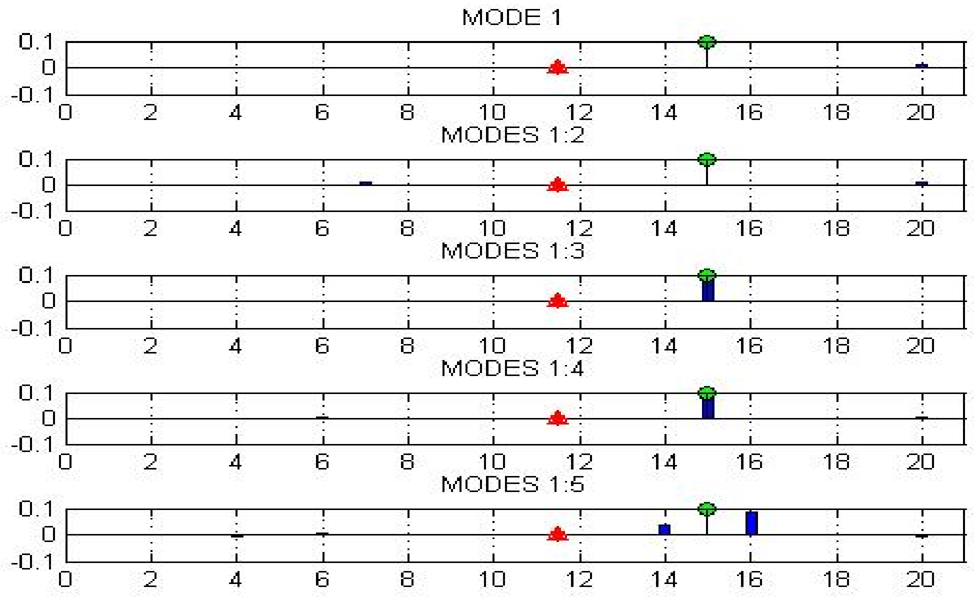

Figure 61. Mass error prediction Case 6.

D. STIFFNESS DAMAGE DETECTION WITH MULTIPLE ABC’S

With the usefulness of the ABC’s proven for a single pinned ABC and one single damaged element, and using as guidance the different patterns found according to the different configuration of the ABC’s; it is inferred that by the use of more than one ABC, the system is going to become less underdetermined, therefore it is going to be able to better predict the error perturbations imposed in the cantilever beam model.

The main goal is to configure the minimum number of ABC’s such that high enough sensitivity levels are attained throughout the structure, hence being able to have reliable damage detection capability at all times and at any beam location, considering that in a real life structure the location of the damaged elements are very likely to be unknown. This goal is going to be explored by the use of three and five ABC’s configurations along the cantilever beam, for one, two, and four damaged elements in order to test the ability of predicting the different stiffness errors in a reliable form.

The following Figures are intended to represent damage detection for a different number and locations of elements with an imposed stiffness error, by the use of multiple ABC’s along the modeled structure. The sensitivity distribution graphs for the following scenarios were omitted, since it is already understood that the sensitivity levels are going to get increased a those DOF’s where the pinned ABC’s are applied.

The pinned ABC’s are represented by the red triangles, the elements with the stiffness perturbations are represented in brown color. The stem plots with the blue circles on the top indicate the magnitude and localization of the actual stiffness errors.

1. One Stiffness Damaged Element with Three ABC’s

Figure 62 shows the cantilever beam representation for 10 percent stiffness damage at element ten, with a three ABC’s set up applied at DOF’s 15, 29, and 41:

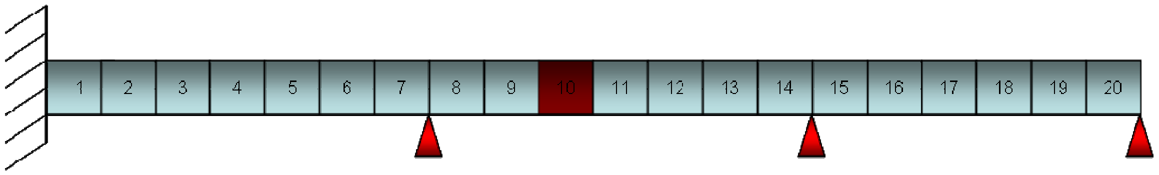

Figure 62. Stiffness error at element 10, ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

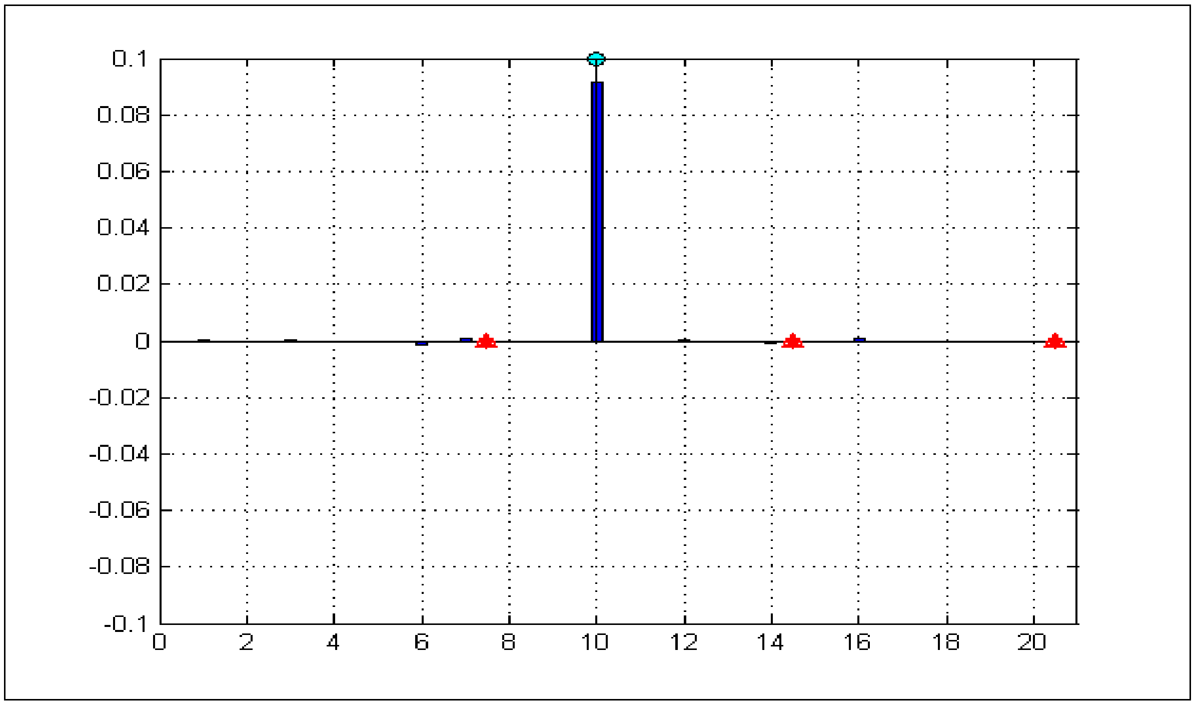

Figure 63. Stiffness error prediction at element 10 with 3 ABC's.

Figure 64 shows the cantilever beam representation for a 10 percent stiffness damage at element five, and with the same three ABC's set up:

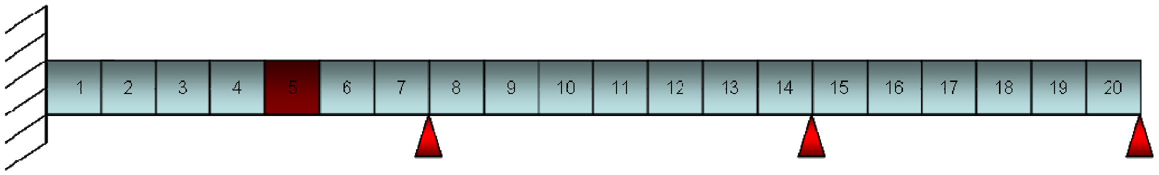

Figure 64. Stiffness error at element 5. ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

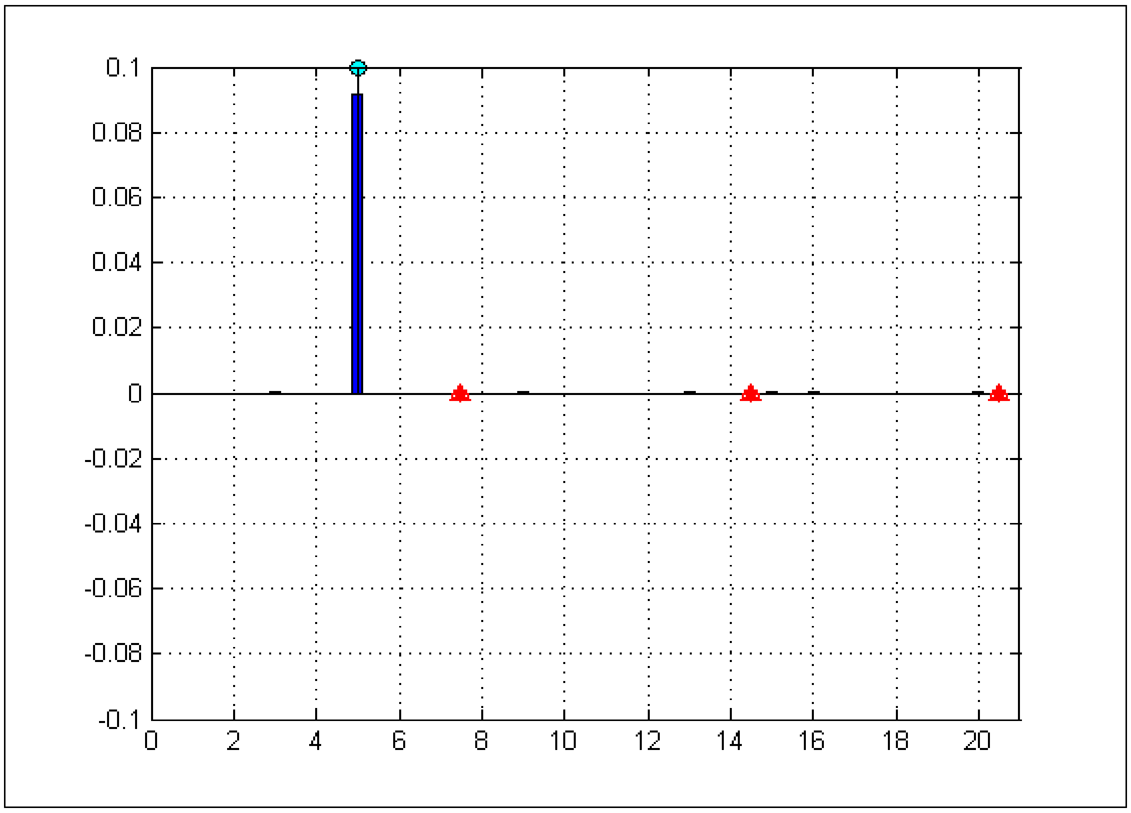

Figure 65. Stiffness error prediction at element 5 with 3 ABC's.

2. Two Stiffness Damaged Elements with Three ABC's

Figure 66 depicts the same three ABC configuration, but now for a 10 percent stiffness damaged at elements 5 and 18:

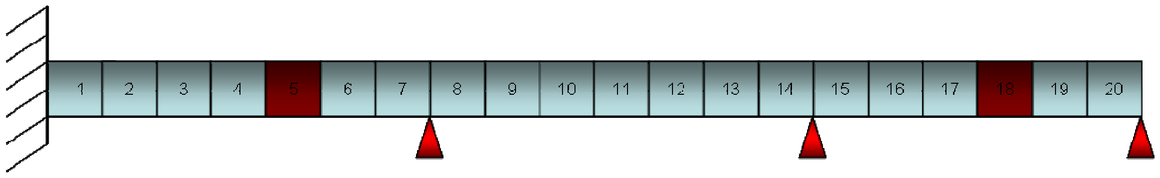

Figure 66. Stiffness error at elements 5, and 18. ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

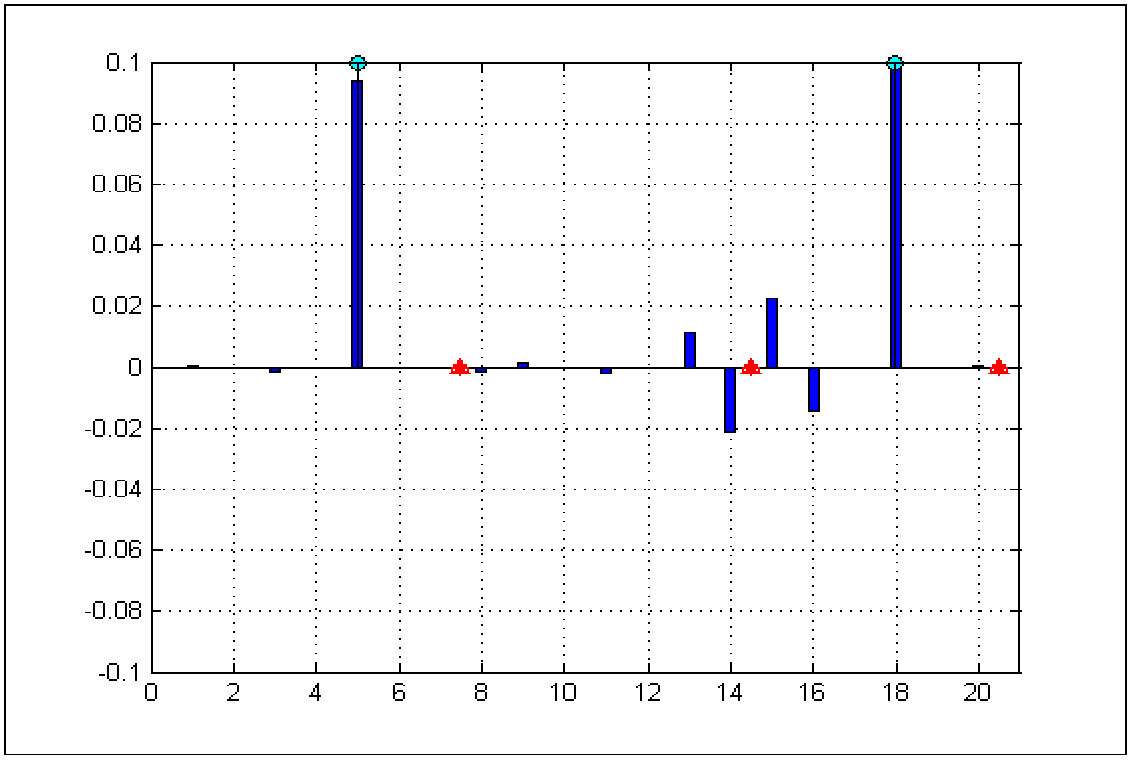

Figure 67. Stiffness error prediction at elements 5, and 18. 3 ABC's.

3. Four Stiffness Damaged Elements with Three ABC's

Figure 68 depicts the standard three ABC's set up chosen for these experiments, for a 10 percent stiffness damage at elements 5, 10, 15, and 20:

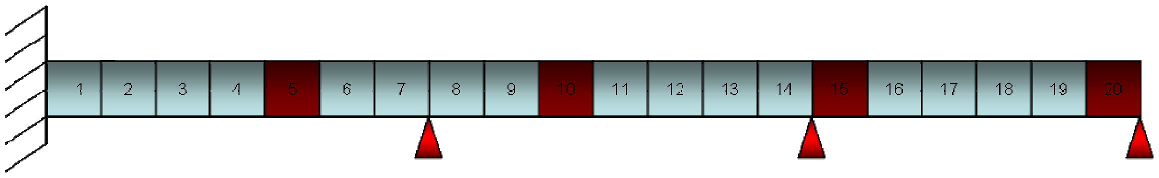

Figure 68. Stiffness error at elements 5, 10, 15, and 20. ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

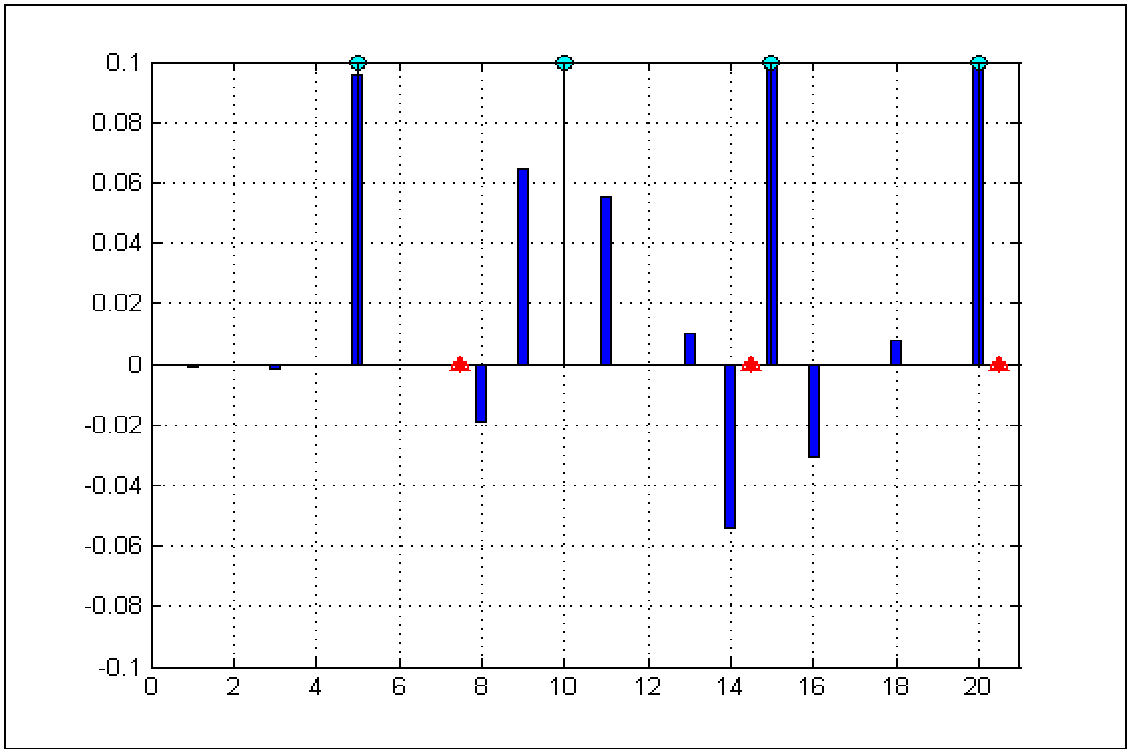

Figure 69. Stiffness error prediction at elements 5, 10, 15, and 20. 3 ABC's.

4. One Stiffness Damaged Element with Five ABC's

Figure 70 shows the cantilever beam representation for 10 percent stiffness damage at element ten, with a five ABC's set up applied at DOF's 7, 15, 23, 33 and 41:

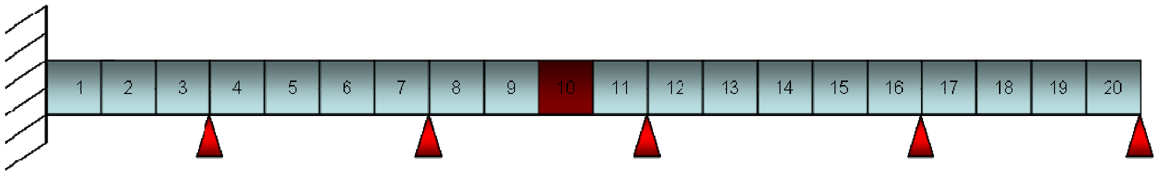

Figure 70. Stiffness error at element 10. ABC's pinned at DOF's 7, 15, 23, 33, 41.

The error prediction for the above configuration is the following:

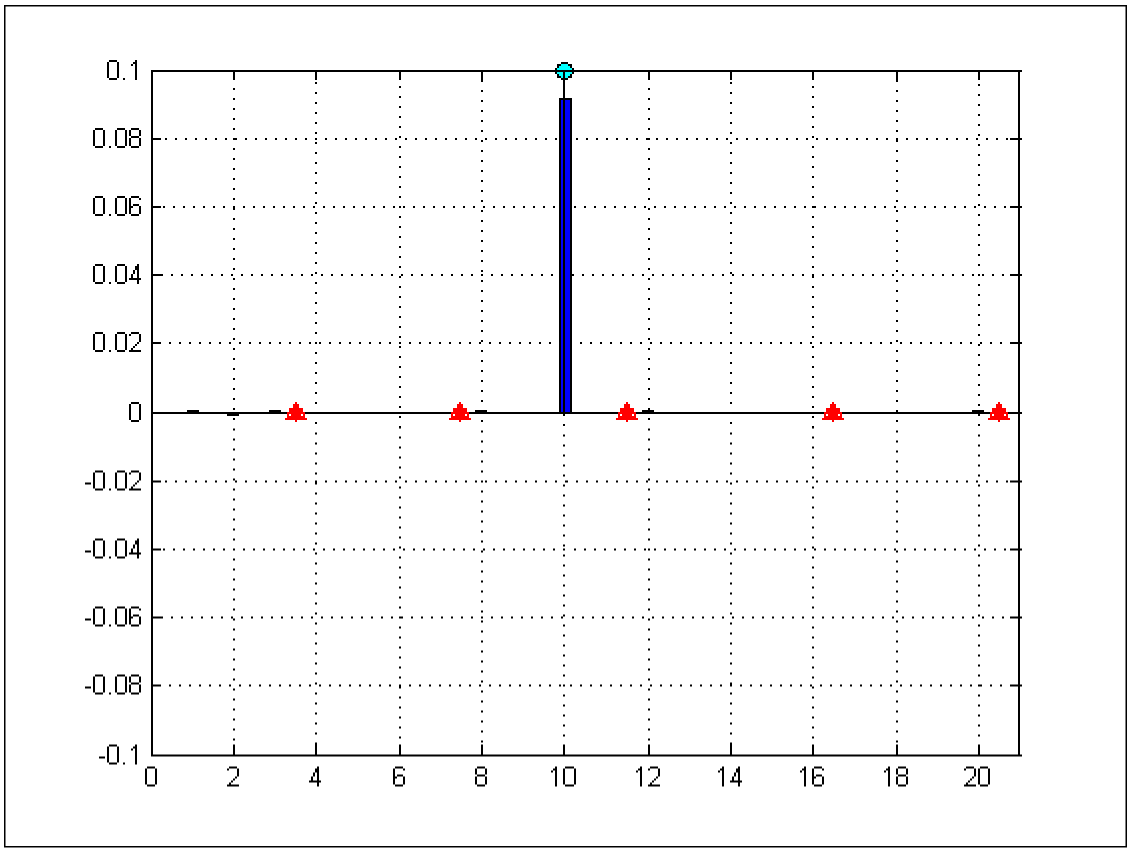

Figure 71. Stiffness error prediction at element 10. 5 ABC's.

Figure 72 shows the cantilever beam representation for a 10 percent stiffness damage at element five, and with the same five ABC's configuration:

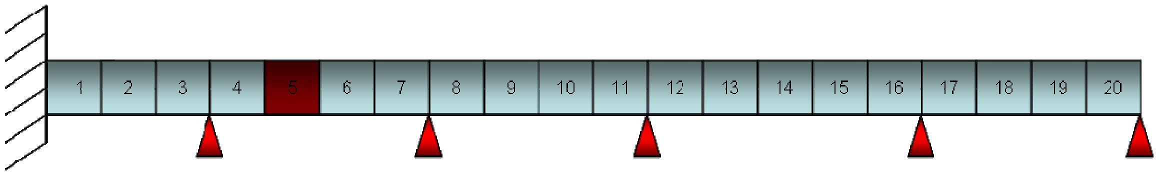

Figure 72. Stiffness error at element 5. ABC's pinned at DOF's 7, 15, 23, 33, 41.

The error prediction for the above configuration is the following:

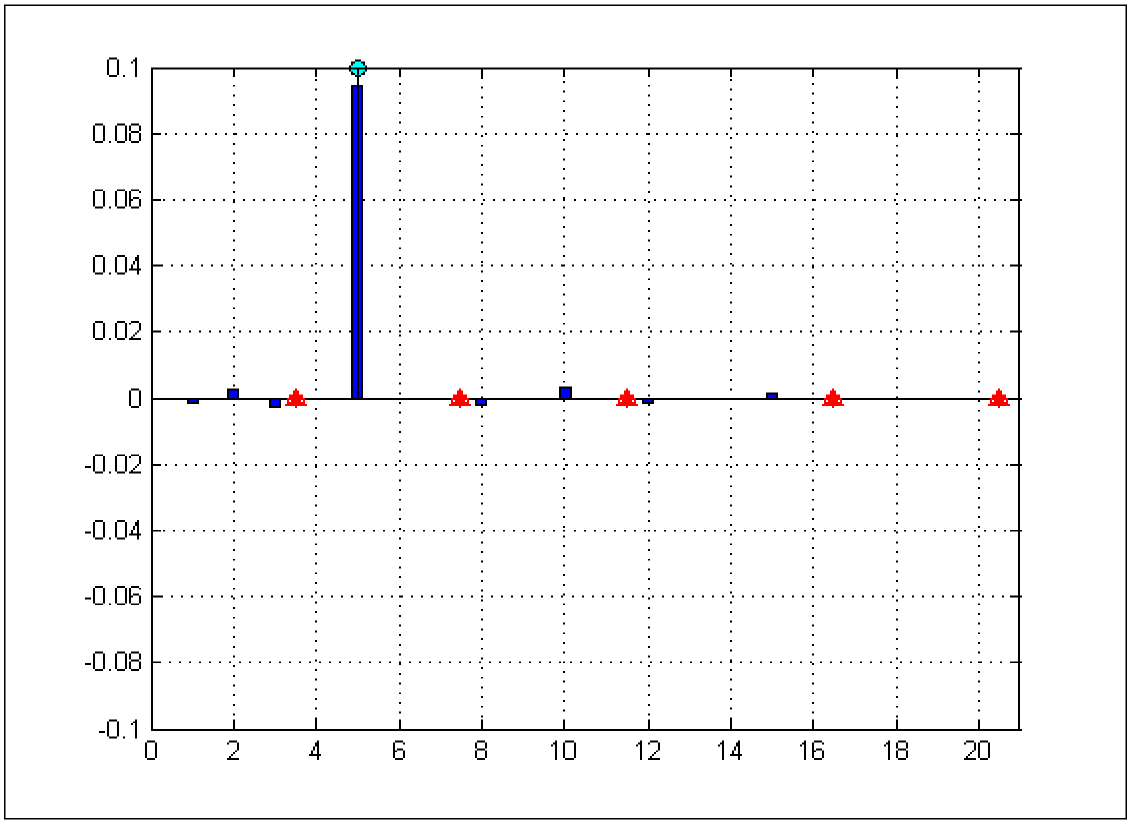

Figure 73. Stiffness error prediction at element 5. 5 ABC's.

5. Two Stiffness Damaged Elements with Five ABC's

Figure 74 depicts the five ABC's set up, for a 10 percent stiffness damaged at elements 5 and 18:

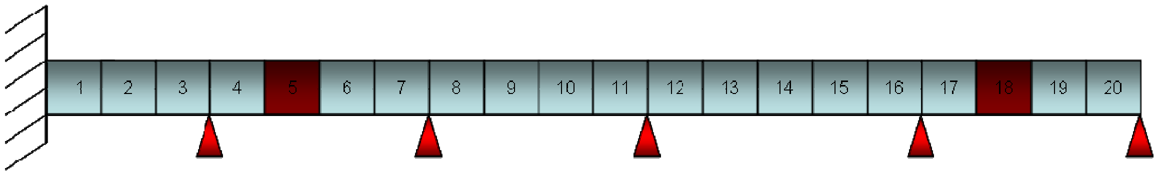

Figure 74. Stiffness error at elements 5 and 18. ABC's pinned at DOF's 7, 15, 23, 33, 41.

The error prediction for the above configuration is the following:

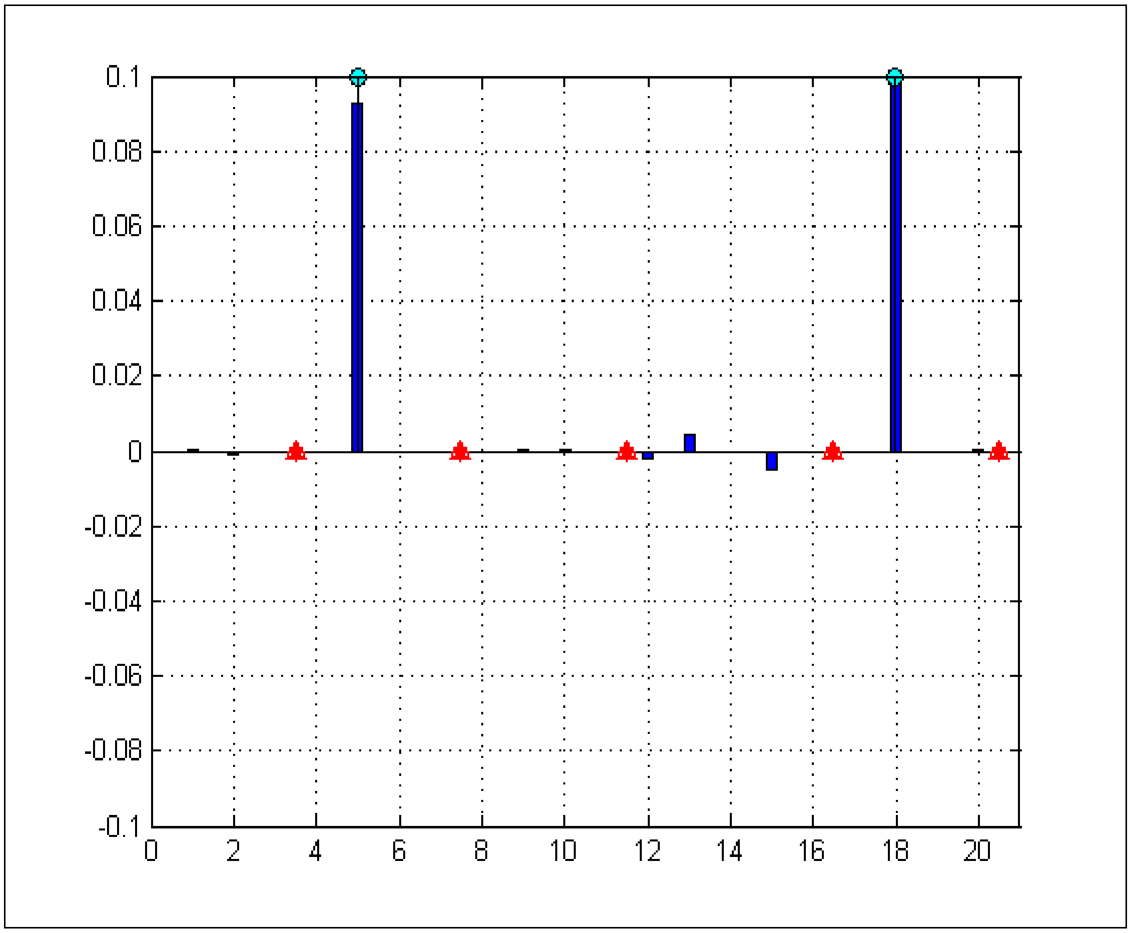

Figure 75. Stiffness error prediction at elements 5 and 18. 5 ABC's.

6. Four Stiffness Damaged Elements with Five ABC's

Figure 76 depicts the five ABC's set up, but now for a 10 percent stiffness damage at elements 5, 10, 15, and 20:

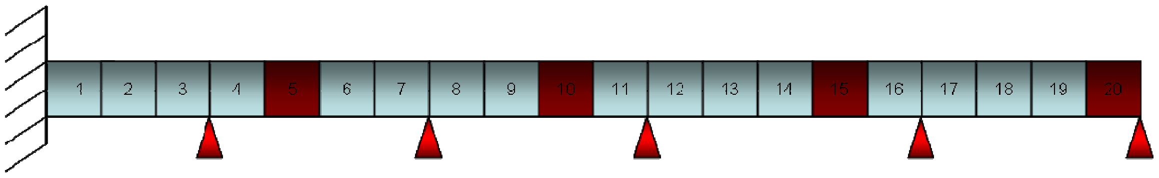

Figure 76. Stiffness error at elements 5, 10, 15, and 20. ABC's pinned at DOF's 7, 15, 23, 33, 41.

The error prediction for the above configuration is the following:

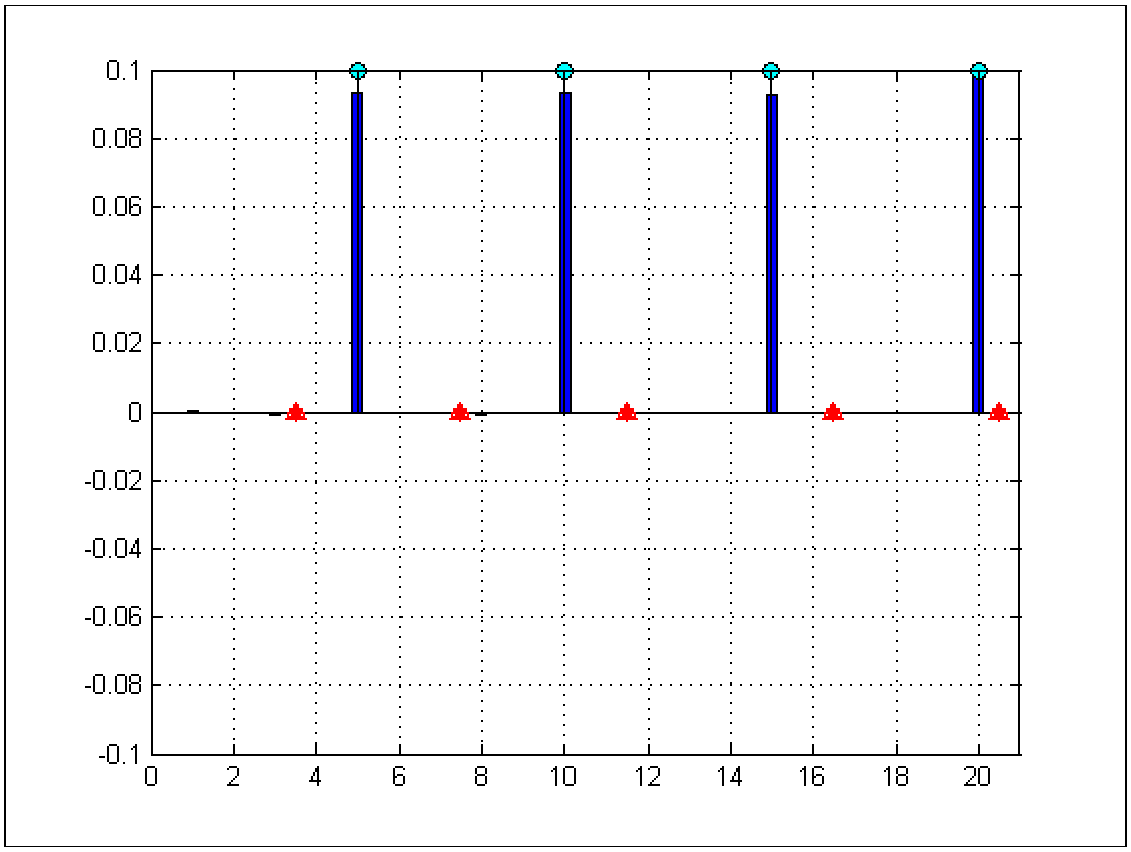

Figure 77. Stiffness error prediction at elements 5, 10, 15, and 20. 5 ABC's.

E. MASS DAMAGE DETECTION WITH MULTIPLE ABC'S

The following six scenarios represent the damage detection capabilities of multiple ABC's for one, two, and four mass damaged elements. In the following Figures the pinned ABC's are represented by the red triangles, the elements with the mass perturbations are represented in magenta color. The stem plots with the yellow circles on the top indicate the magnitude and localization of the actual mass errors.

1. One Mass Damaged Element with Three ABC's

Figure 78 shows the cantilever beam representation for 10 percent mass damage at element eight, with a three ABC's configuration applied at DOF's 15, 29, and 41:

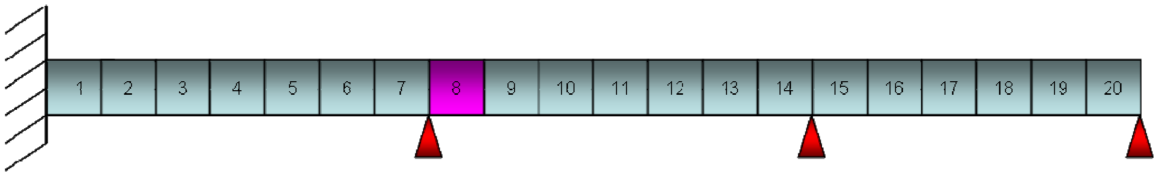

Figure 78. Mass error at element 8, ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

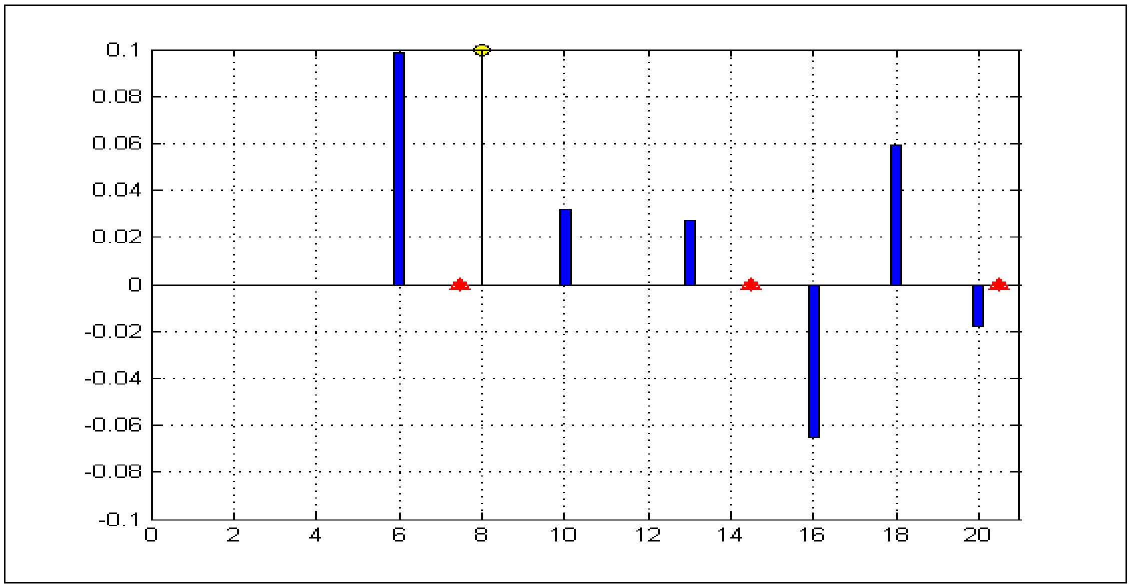

Figure 79. Mass error prediction at element 8. 3 ABC's.

2. Two Mass Damaged Elements with Three ABC's

Figure 80 shows the cantilever beam representation for 10 percent mass damage at elements five and fifteen, with a three ABC's set up applied at DOF's 15, 29, and 41:

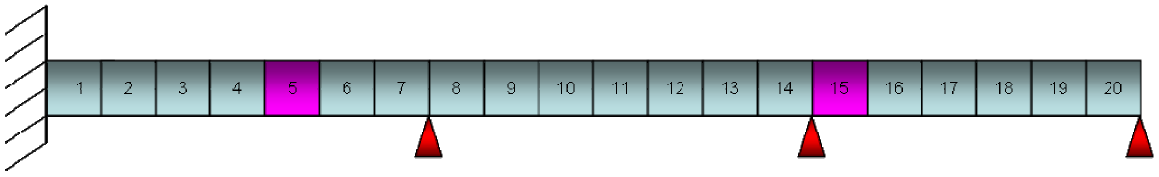

Figure 80. Mass errors at elements 5 and 15, ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

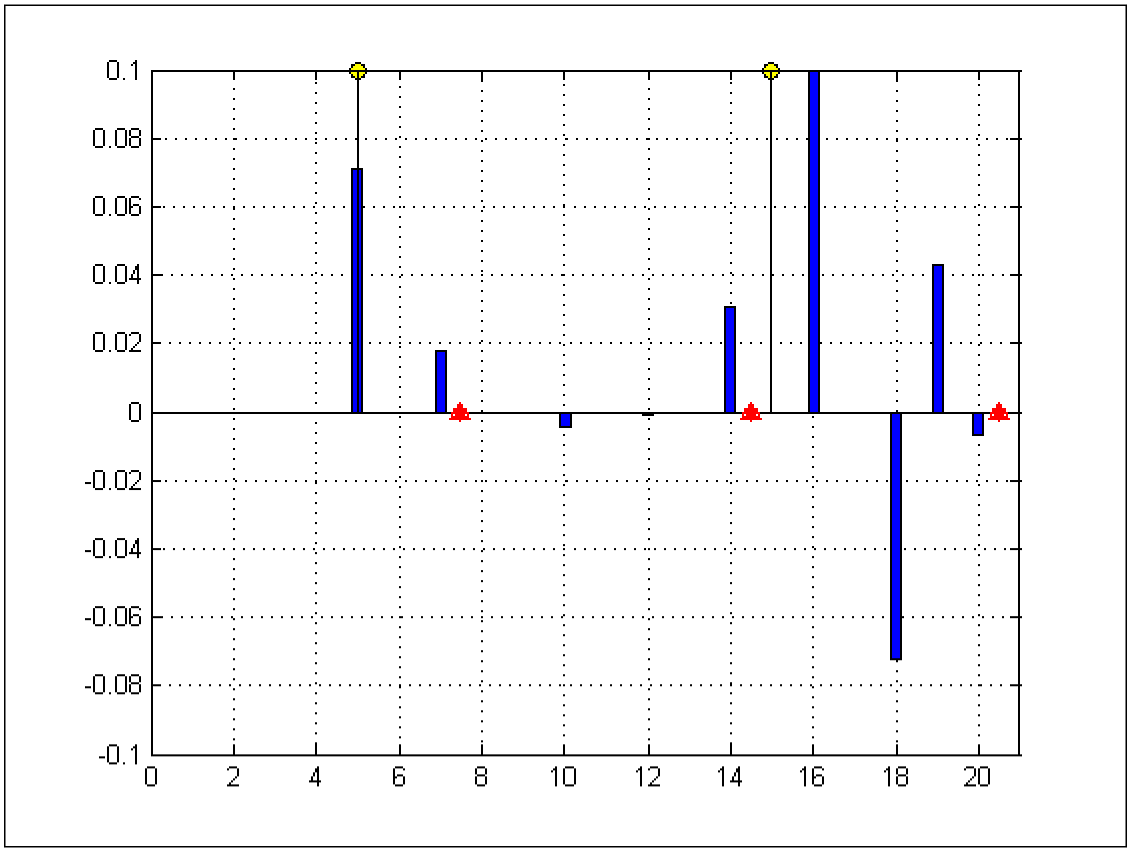

Figure 81. Mass errors prediction at elements 5 and 15. 3 ABC's.

Figure 82 shows the cantilever beam representation for 10 percent mass damage at elements ten and eighteen, with a three ABC's set up applied at DOF's 15, 29, and 41:

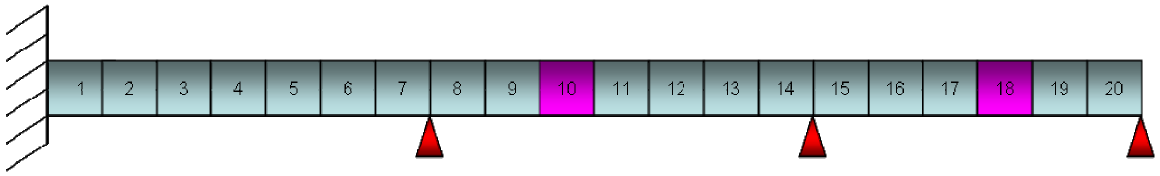

Figure 82. Mass errors at elements 10 and 18, ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

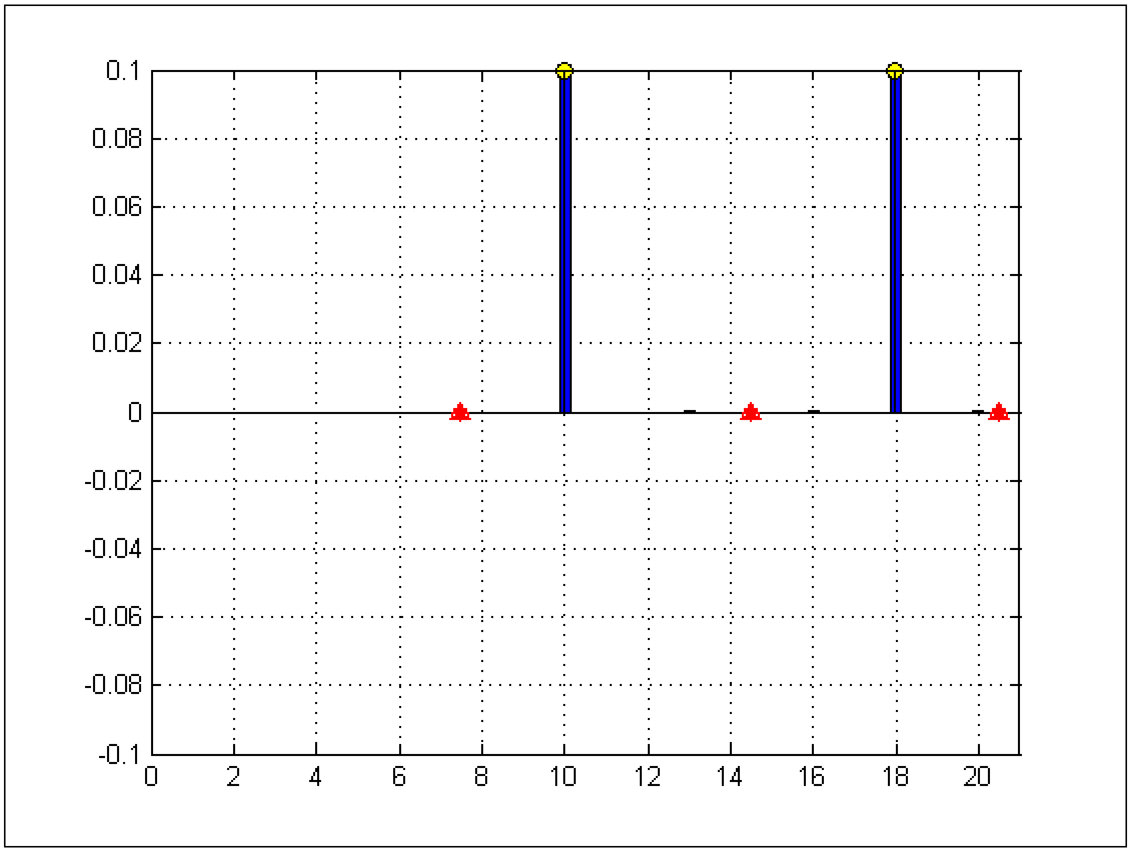

Figure 83. Mass errors prediction at elements 10 and 18. 3 ABC's.

3. Four Mass Damaged Elements with Three ABC's

Figure 84 shows the cantilever beam representation for 10 percent mass damage at elements five, ten, fifteen, and twenty, with a three ABC's configuration applied at DOF's 15, 29, and 41:

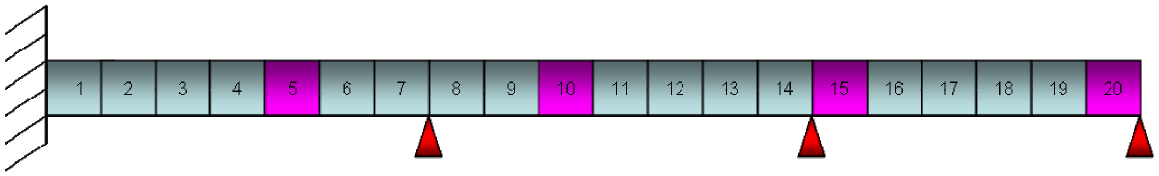

Figure 84. Mass errors at elements 5, 10, 15, and 20. ABC's pinned at DOF's 15, 29, 41.

The error prediction for the above configuration is the following:

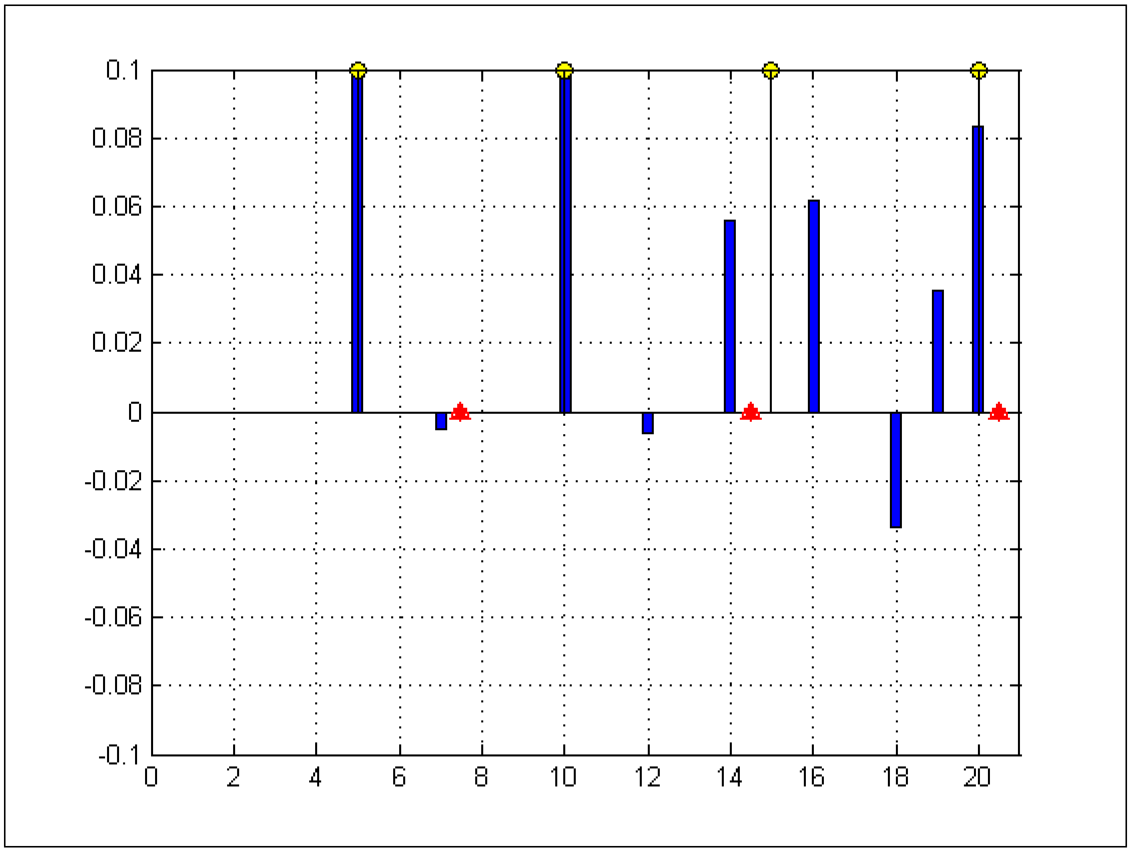

Figure 85. Mass errors prediction at elements 5, 10, 15, and 20. 3 ABC's.

4. One Mass Damaged Element with Five ABC's

Figure 86 shows the cantilever beam representation for 10 percent mass damage at element eight, with a five ABC's configuration applied at DOF's 7, 15, 23, 33, and 41:

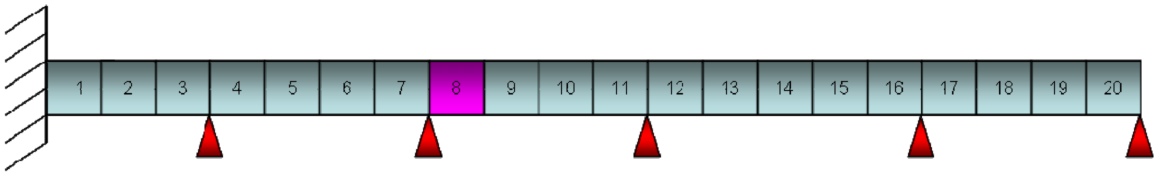

Figure 86. Mass error at element 8. ABC's pinned at DOF's 7, 15, 23, 33, and 41.

The error prediction for the above configuration is the following:

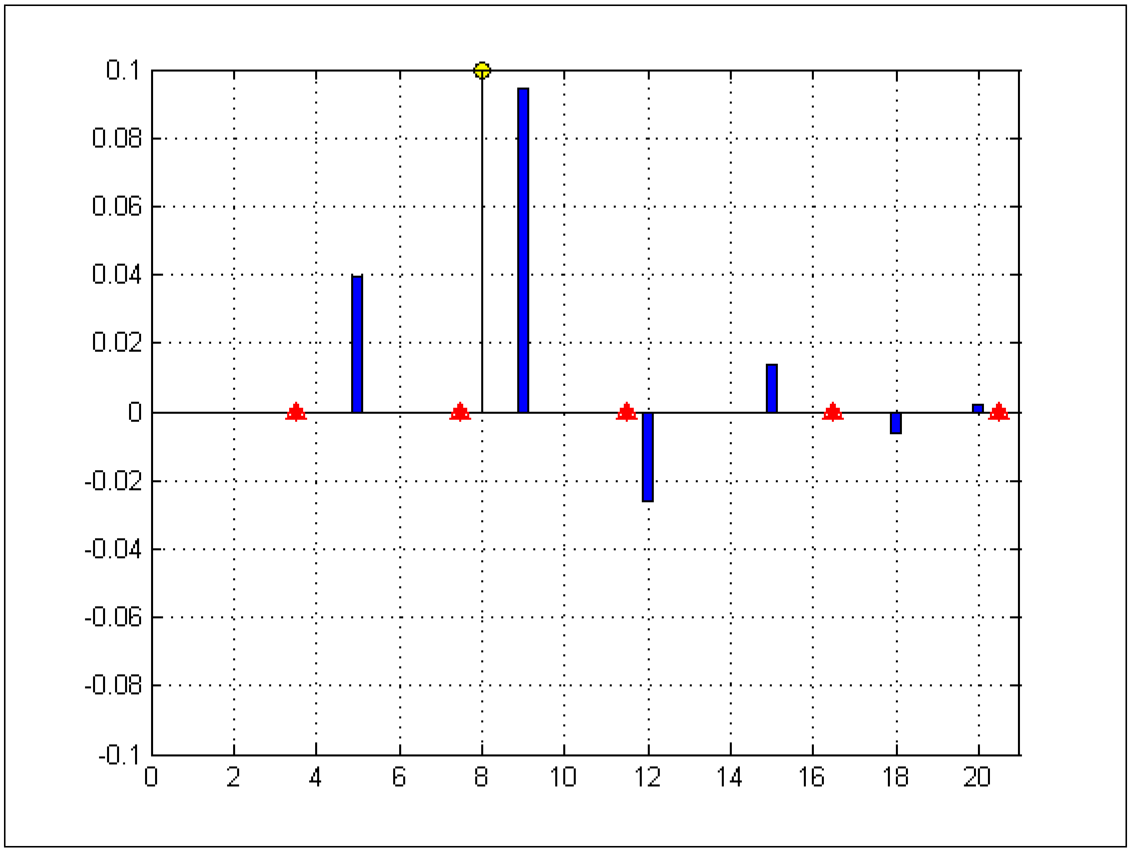

Figure 87. Mass error prediction at element 8. 5 ABC's.

5. Two Mass Damaged Elements with Five ABC's

Figure 88 shows the cantilever beam representation for 10 percent mass damage at elements seven and eleven, with a five ABC configuration applied at DOF's 7, 15, 23, 33, and 41:

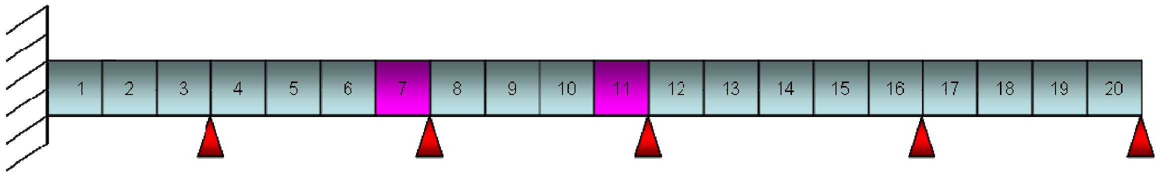

Figure 88. Mass error at elements 7 and 11 ABC's pinned at DOF's 7, 15, 23, 33, and 41.

The error prediction for the above configuration is the following:

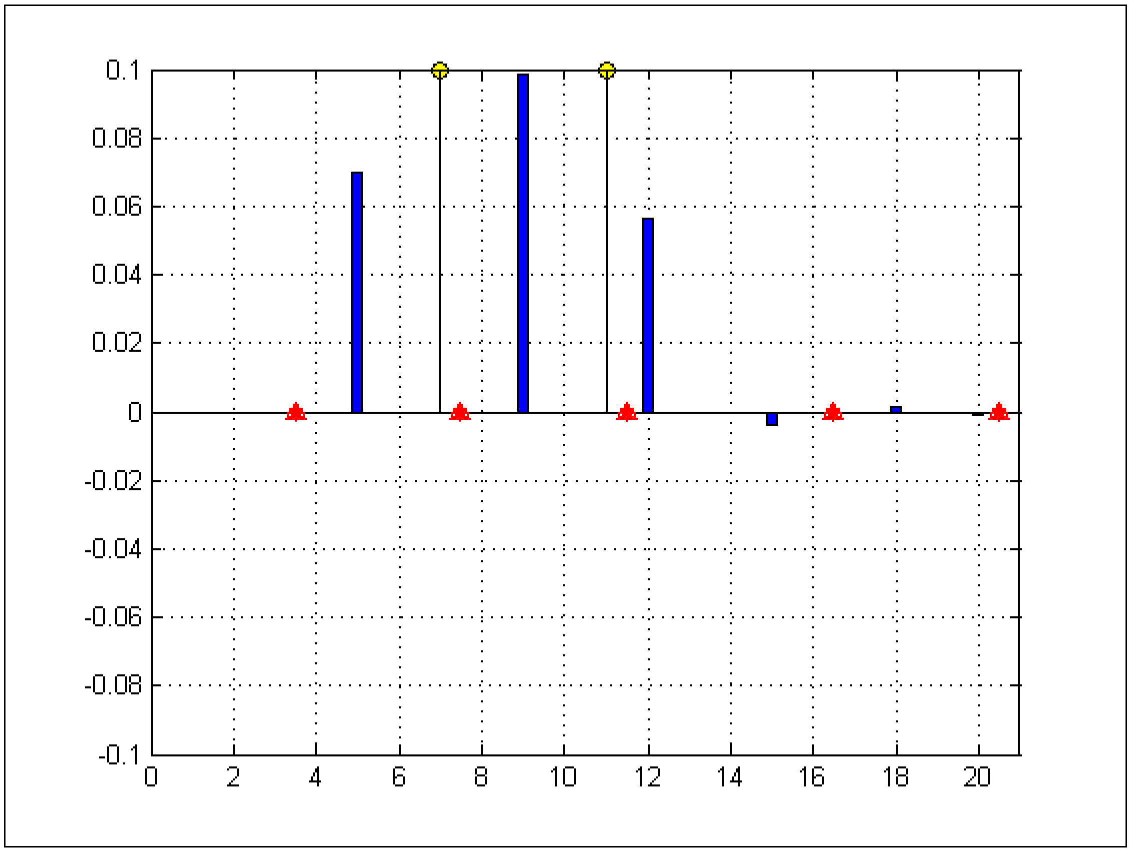

Figure 89. Mass error prediction at elements 7 and 11. 5 ABC's.

Figure 90 shows the cantilever beam representation for 10 percent mass damage at elements five and nine, with a five ABC's set up applied at DOF's 7, 15, 23, 33, and 41:

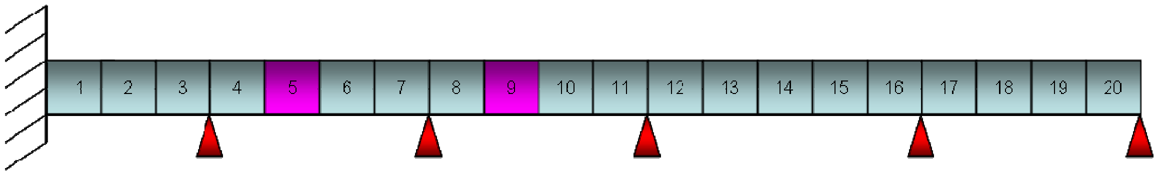

Figure 90. Mass error at elements 5 and 9. ABC's pinned at DOF's 7, 15, 23, 33, and 41.

The error prediction for the above configuration is the following:

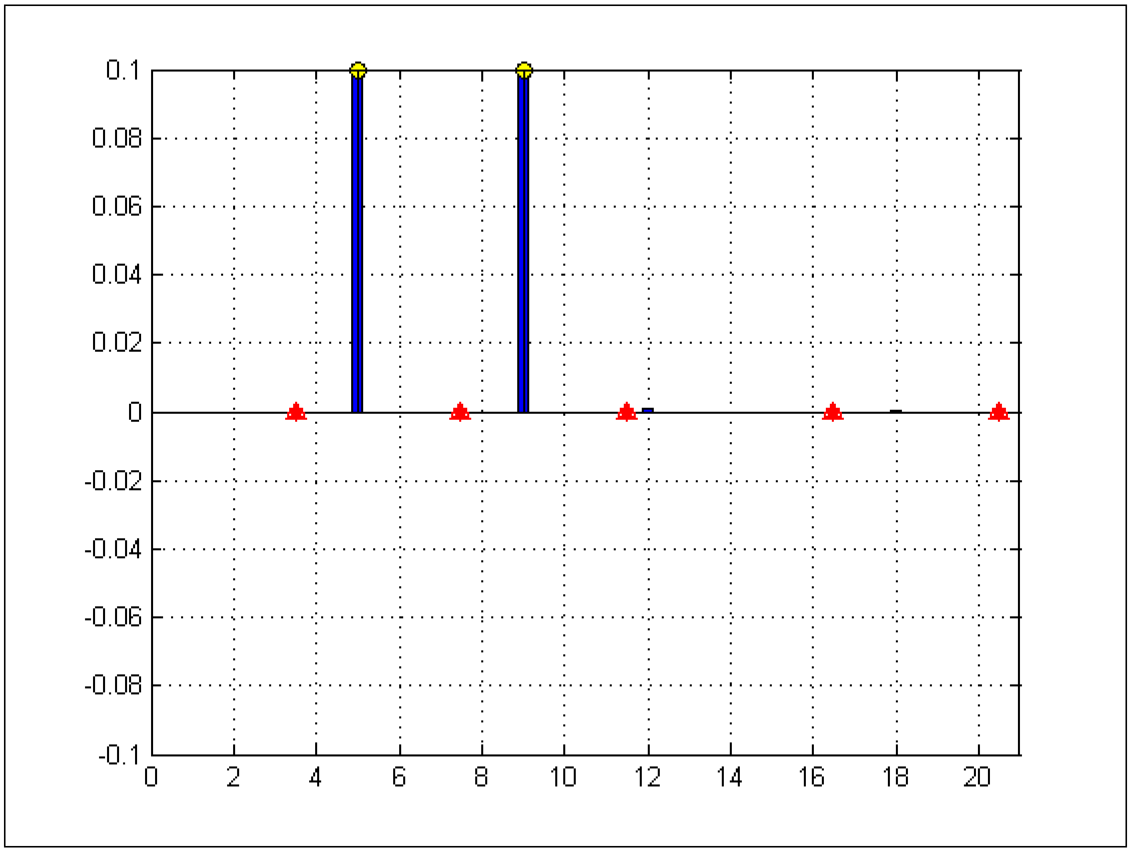

Figure 91. Mass error prediction at elements 5 and 9. 5 ABC's.

6. Four Mass Damaged Elements with Five ABC's

Figure 92 shows the cantilever beam representation for 10 percent mass damage at elements three, seven, eleven, and seventeen, with a five ABC's set up applied at DOF's 7, 15, 23, 33, and 41:

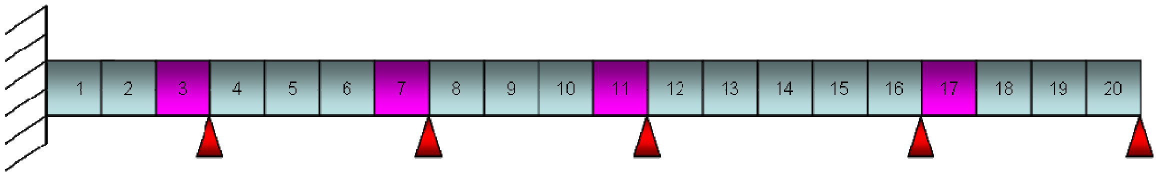

Figure 92. Mass error at elements 3, 7, 11, and 17. ABC's pinned at DOF's 7, 15, 23, 33, and 41.

The error prediction for the above configuration is the following:

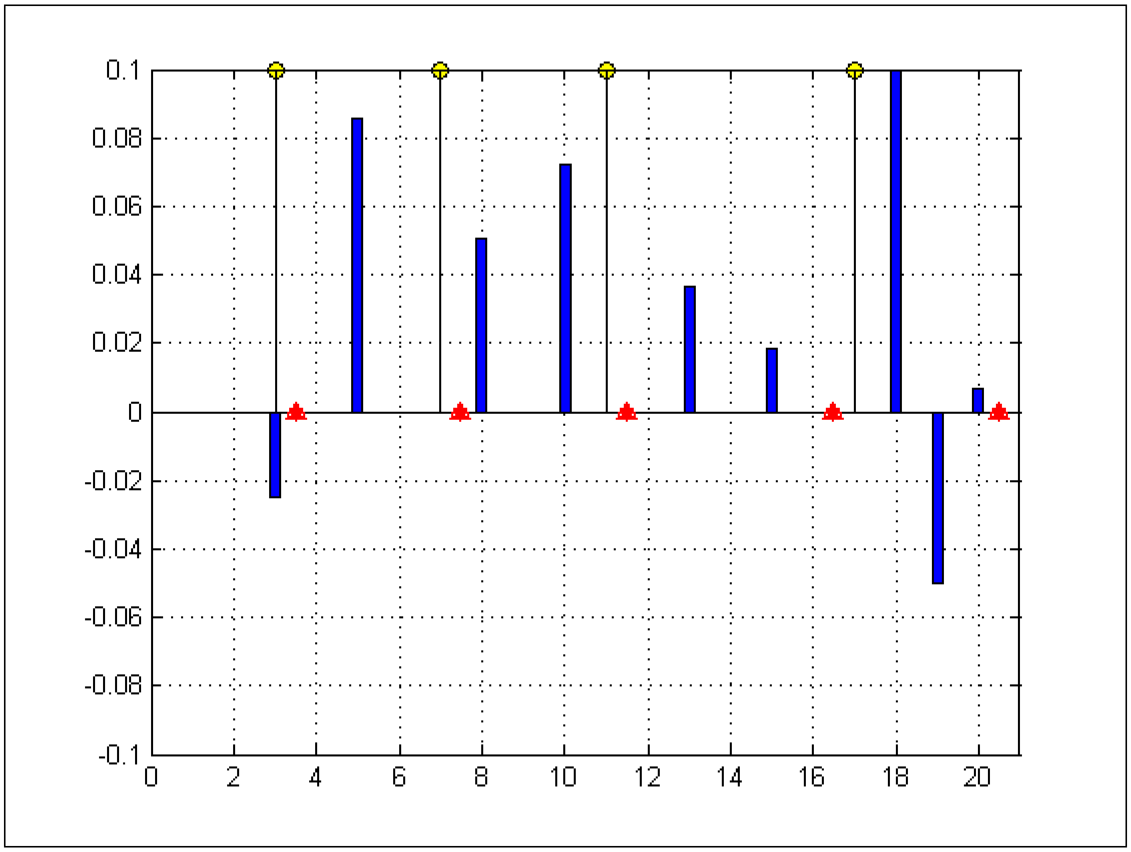

Figure 93. Mass error prediction at elements 3, 7, 11, and 17. 5 ABC's.

Figure 94 shows the cantilever beam representation for 10 percent mass damage at elements five, nine, fifteen and eighteen, with a five ABC's set up applied at DOF's 7, 15, 23, 33, and 41:

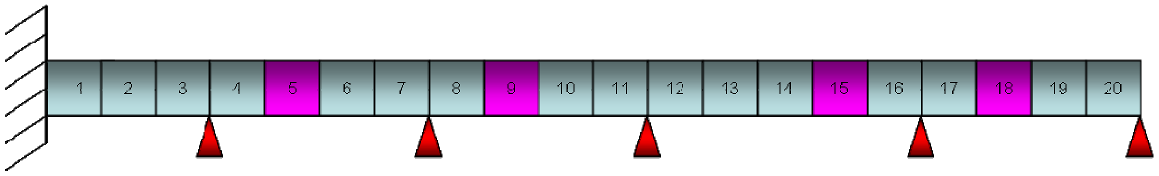

Figure 94. Mass error at elements5, 9, 15, and 18. ABC's pinned at DOF's 7, 15, 23, 33, and 41.

The error prediction for the above configuration is the following:

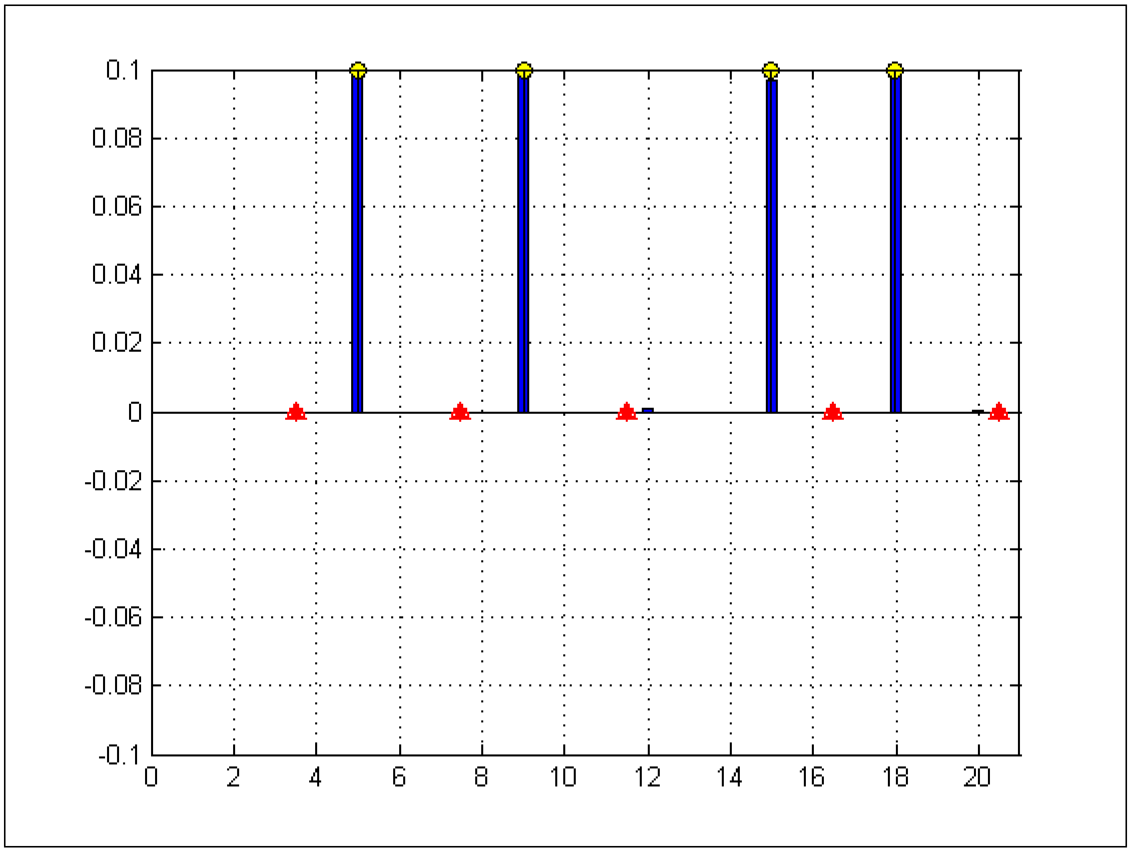

Figure 95. Mass error prediction at elements 5, 9, 15, and 18. 5 ABC's.

F. DISCUSSION OF RESULTS

The sensitivity of the natural frequencies of the structure to changes in stiffness and mass were calculated from the finite element analysis.

The twelve cases analyzed for the changes in stiffness; six for one ABC and one damaged element, and six for multiple ABC's and multiple damaged number of elements, demonstrated a very specific behavior.

For the one ABC and one damaged element cases, the highest probability of detecting the damage proved to be when the stiffness error was the closest to the fixed end of the cantilever beam, which happens to be the highest exposed part to stiffness failure due to the increased stress concentration in that area. The further the stiffness error perturbation was translated to the free end, the less likely it was detected, hence the use of the pinned ABC at a DOF adjacent to the element in error, highly increased the possibilities of detecting the damage.

When the pinned ABC was applied to a DOF further away with respect to the location of the damage element, the error was never found, unless it was located very close to the fixed end of the cantilever beam, where the Stiffness Base sensitivity assisted in the detection of it.

On the other hand, the multiple ABC and multiple damaged number of elements cases, significantly improved the damage detection capacity. For the 3 ABC's configuration, the damage was successfully found up to two damaged elements, when four damaged elements were introduced, the system accurately predicted three out of the four damaged elements, and the fourth element in error although it was not detected, was very well bounded, as it is depicted in Figure 69. The five ABC configuration successfully found the damage imposed over the elements in error. Further experiments (not depicted in the present thesis) with more than only four damaged elements successfully retained the errors imposed over the system structure; this explains how due to the stable behavior of the ABC's when to stiffness errors is concerned and making the problem less underdetermined with the incorporation of more ABC's, the error is very likely to be found.

The twelve cases analyzed for the changes in mass; six for one ABC and one damaged element, and six for multiple ABC's and multiple damaged number of elements, did not prove the same consistencies as the ones found in the stiffness errors.

Since the mass errors are related to the inertia of the system, the pinned ABC's were set up in such an approach that were able to produce a considerable displacement at the location of the damaged element, nevertheless, while this reflection holds true, not all of the different pinned ABC combinations at those DOF' that produced higher kinetic energy seemed to detect the mass error perturbation successfully.

For the one ABC and one damaged element cases, when the pinned ABC was set up to emulate the beam model as if it were a clamped-clamped beam, the error was sparsely detected and with no pattern defined; but the best error predictions were found when the ABC was configured to emulate the beam model as a cantilever beam, exerting the maximum inertia (displacement) at the damaged element.

On regard to the multiple ABC's and multiple damaged number of elements cases, the pattern found for the one ABC and one mass damage cases, stills holds the same. It was discovered that even when the configuration of more ABC helped to reduce the undetermined nature of the finite element problem, therefore being able to predict a few more elements in error, no specific pattern was found. Nevertheless it was recognized that the mass errors are a very strong function of the kinetic energy of the system. Figures 81, 85, 87, 89, and 93, show how the errors located at low kinetic energy regions were not detected, whereas, Figures 83, 91, and 95 show how the errors were accurately predicted when a smart combination of ABC produced areas of high kinetic energy at the regions where the damaged elements were located.

On the following chapter an Optimization approach is develop in order to find the mass error in a cantilever beam. As it will be seen, the MATLAB built in optimization function, proves to work as a reliable technique.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. OPTIMIZATION APPROACH

The purpose of numerical optimization is to aid in rationally searching for the design variables to satisfy constraints (Vanderplaats, 1984). That is, to match the dynamic response of a finite element model to that of the experimental response of the system of interest.

An optimization program was developed to predict the mass error detection/prediction on the cantilever beam model.

A. OPTIMIZATION THEORY

The general problem statement for a non-linear constrained optimization problem is:

To minimize $f(x)$ Objective Function

Subject to:

$$g_j(x) \le 0 \tag{7.1}$$

where $j = 1, m$ Inequality Constraints

$$h_k = 0 \tag{7.2}$$

where $k = 1, p$ Equality Constraints

$$x_i^1 \le x_i \le x_i^u \tag{7.3}$$

where $i = 1, n$ Side Constraints

The design variables vector is represented:

$$x = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{Bmatrix} \tag{7.4}$$

The constraint equations are used to bound possible solution combinations to meet the requirements of the given situation. Most optimization algorithms require that an initial set of design variables, x^0 be specified. Beginning from this starting point, the design is updated iteratively. This process can be written as:

$$x^q = x^{q-1} + \alpha s^q \tag{7.5}$$

where q represents the iteration number and s is a vector search direction in the design space. The scalar quantity α defines the distance that we wish to move in direction s. The updated values of x are used to calculate the value of the objective function for each iteration. The search direction is chosen to decrease the value of the objective function while staying within the constraints of the system. The process continues until the objective function converges to a local minimum and the optimal solution is localized.

1. Modal Assurance Criterion (MAC)

The modal assurance criterion is a scalar constant relating the causal relationship between two modal vectors. The MAC is used as a means to compare analytically and experimentally obtained vibrational mode shapes. The MAC will have a value between 0 and 1. A value of 0 indicates that the two modal vectors are not correlated while a value of 1 indicates the modal vectors are correlated. The method of calculating the MAC for comparing the i^{th} mode of the analytical system to the i^{th} model of the experimental system is:

$$MAC = \frac{\left|\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right|^{2}}{\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right)\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right)} \tag{7.6}$$

B. PROBLEM FORMULATION

The optimization problem was analyzed according to the cantilever beam displayed on Figure 96, with the same beam specifications described on Table 1 from Chapter IV. The beam consisted of 42 elements, each 1 inch in length and DOF 1 was at the fixed end of the cantilever beam; this corresponded to FE model element quantity and length. A Series 336 FLEXCEL ICP accelerometer (serial number 10860) was threaded into position at node 42 of the beam and wired into Channel 2 on a DACTRON Focus front end digital signal processor (DSP). An excitation was applied by a PCB Series 086 B03 impact hammer (serial number 269), which was wired into Channel 1 on the DSP. The accelerometer and force hammer were calibrated using a pendulous test; the sensitivity adjustment was applied in the set-up of RT Pro Focus 5.57 software.

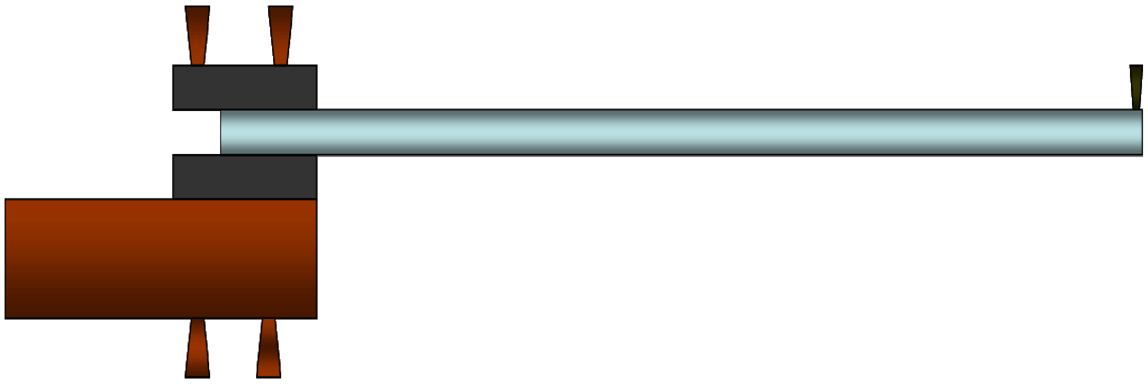
Figure 96. Experimental beam set up.

The experiment was performed in two segments: The analytical and the experimental. In the analytical, the *Build2beams.m* MATLAB code was used to generate the original finite element beam model set of original eigenvalues and eigenvectors, then the damage (mass perturbation) used during the real experiment, was simulated in MATLAB, generating the modified set of eigenvalues and eigenvectors; subsequently these modal parameters were evaluated into the MATLAB optimization program. The mass error perturbation was imposed on element 35 of the cantilever beam, with a 10% mass excess with respect to the total beam mass. The finite element model was analyzed from modes one to eight.

The second (experimental) segment consisted of the actual measured data from the laboratory. It was created after tapping the beam shown in figure 62, producing the corresponding Frequencies Response Forces (FRF) by means of the softwares: DACTRON RT Pro Focus 5.57, and ME'scopeVES; node 42 remained the reference as the load cell roved from one node to the next allowing for the measurement of the response at each node.

The modal data extracted out of the ME'scopeVES software comprises the natural frequencies, damping ratios and magnitudes and phases of the different mode shapes of the described cantilever beam with the mass perturbation added.

1. The RT Pro Focus 5.57 software "Real-time" was configured to measure 3200 spectral lines, 8192 points, with a delta T of 166.7μs over the frequency range 0-2400Hz. The frequency range of 0-2400 Hz was chosen because it covered the first 10 modes of system and signal resolution was sufficient for data acquisition. The excitation signal proved to be clean and thus no window was used for data measuring.

2. Channel set-up

	Channel 1 (Excitation)	Channel 2(Response)
Max Volts (mV):	0.1	0.3
Quantity:	Force	Accel.

EU: lbf   gn

mv/EU: 8.67   104.383

Coupling: ICP AC 0.7 Hz   ICP AC 0.7 Hz

Sensitivity Adjustment: 0   0

3. Trigger set-up

Source: Analog input

Run Mode: Manual Arm every frame

Input: Channel 1

Slope: Bi-polar

Level (%): 1, Level (V): 0

Pre/Post Points (-/+): -10

Pre/Post Time (-/+): -1.67μs

4. Average set-up:

Type: Linear

Domain: Frequency

Frames: 3 (Each node was excited 3 times and an average taken and saved.)

Accept/Reject: Manual Accept/Reject every frame.

(The user rejects double taps, under powered or overloaded signals.)

5. Modal Coordinate set-up:

Auto increment: ON

Rove: Excitation

Point increment: 1

Export: UFF text format, frequency response.

Once the experimental modal parameters were extracted, a Complex to Real Conversion was performed on the mode shapes. Complex ones, as derived from analysis of test data, and real ones, such as are traditionally produced from theoretical analyses in the absence of a detailed knowledge of the nature, extent and distribution of damping (Ewins, 2000). The corresponding information was exported to MATLAB as a ".txt file" such that the corresponding experimental eigenvalues and eigenvectors could be read in a matrix form.

The modal to spatial transformation was obtained by the use of the "Simple Method" (Ewins, 2000); through this method the modulus and the phase of the modal eigenvectors were obtained from the MEscopesVES software. The modulus being interpreted as the square root of the sum of the squares of the real and imaginary parts of the modal analysis data, and the phase was assigned to each node either as 0 or 180 degrees, which represented the positive or negative value of the displacement part of the experimental eigenvector matrix. This process was performed within the MEscopeVES software once the shapes table was generated, by selecting "Complexity plot" from the "Display" drop down menu, obtaining a 'Starbust plot' – Argand Diagram (Ewins, 2000), then the modulus and phase table was automatically obtained after clicking on the "Normalize Data" button, now being ready to import the table as the ".txt file" into matlab. Figure 97 depicts the Starbust plot obtained from the 10% mass error exerted at element 35 of the cantilever beam.

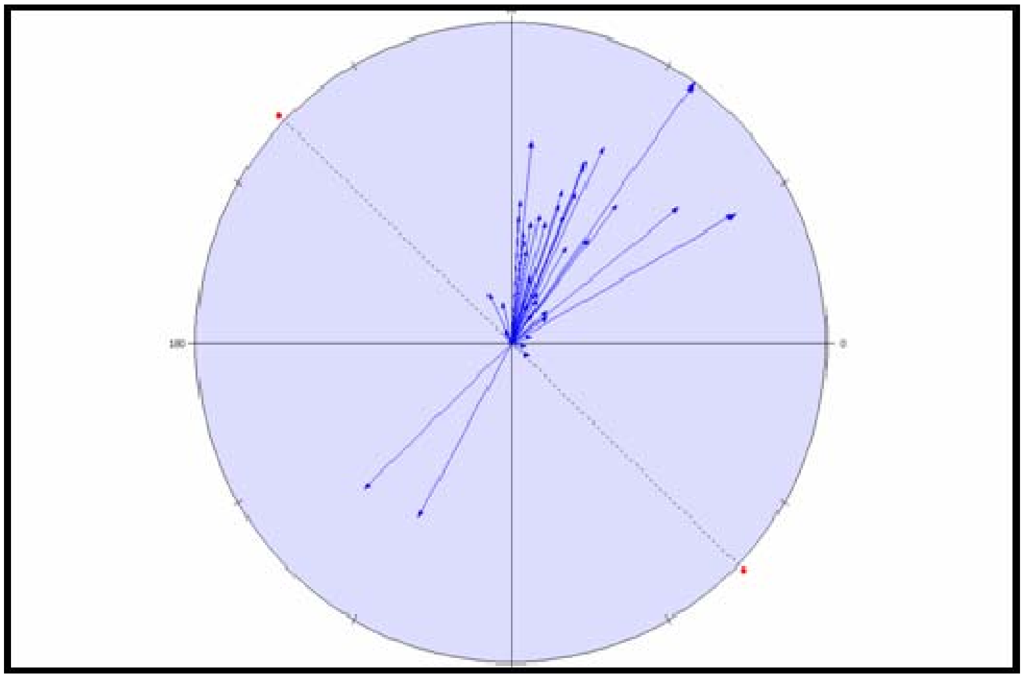

Figure 97. Starbust Plot- Argand Diagram.

C. OPTIMIZATION APPROACH VIA MATLAB

The actual optimization program was run with the use of a built in MATLAB function named "fmincon" which permitted the finding of the error perturbation on the beam. "fmincon" aids in finding a *constrained minimum* of a given Objective Function of several design variables starting at an initial estimate (xo). The fmincon algorithm employs the SQP implementation, which consists of three stages. The first stage updates the Hessian matrix via the quasi- Newton method, where the Hessian matrix is calculated using the Broydon- Fletcher Goldgarb-Shanno (BFGS) method, which is the most used with inexact line searches. The second stage performs the Quadratic Programming Solution, used to calculate the parameter changes, by the calculation of a feasible point and then the generation of an iterative sequence of feasible points that converge to the solution. The last stage essentially performs the line search and generates the merit function. The "fmincon" MATLAB set up used was: *x = fmincon (fun,xo,A,b,Aeq,beq,lb,ub)*

The “fmincon” needs as input arguments the following:

- Objective Function *(fun)*
- Starting Point *(xo)*
- Design Variable (DV)
- Equality Constraints (*Aeq,beq)*
- Inequality Constraints (*A,b and lb,ub)*

1. Objective Function (OF)

In order to prove the certainty and performance of the optimization program, in the mass- error detection analysis, three types of OF’s were implemented:

a. Eigenvector Objective Function

Taken from the Modal Assurance Criterion concept, which quantitatively checks the correlation between the measured and the calculated mode shapes, its arguments lie between 0 and 1; those elements with a value close to 1 indicate that the two modes under comparison are quite close. Thus the MAC was subtracted from one to produce a result that would indicate a better performance when minimized.

$$J(\varphi) = \sum_{l=1}^{p} \left(1 - \frac{\left| \sum_{l=1}^{p} \left\{ \Phi_{l,i}^{e} \right\}^{T} \left\{ \Phi_{l,j}^{a} \right\} \right|^{2}}{\left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^{e} \right\}^{T} \left\{ \Phi_{l,i}^{e} \right\} \right) \left(\sum_{l=1}^{p} \left\{ \Phi_{l,j}^{a} \right\}^{T} \left\{ \Phi_{l,j}^{a} \right\} \right)} \right) \tag{7.7}$$

b. Eigenvalue Objective Function

Natural Frequencies Ratio

$$J(\varphi) = \sum_{l=1}^{p} \left| \frac{\lambda_i^e - \lambda_i^a}{\lambda_i^e} \right| \tag{7.8}$$

c. *Eigenvalue plus Eigenvector Objective Function Combination*

Eigenvalues and eigenvectors were selected as the main core of the OF's since they represent the modal response to the natural properties of the system.

$$J(\varphi) = \sum_{l=1}^{p} \left| \frac{\lambda_l^e - \lambda_l^a}{\lambda_l^e} \right| + \left(1 - \frac{\left| \sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right|^2}{\left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right) \left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right)} \right) \tag{7.9}$$

2. Starting Point (xo)

Since it was always desired to iterate over the complete set of elements, the starting point consisted of a [42x1] unit value vector.

3. Design Variables (DV)

Being the eigenvalues and eigenvectors part of the natural properties of the system and since these are determined by the mass and stiffness of the structure, it was decided to use the density (φ) as the design variable (DV). It was not intended to modify the elasticity modulus (E) of the beam, such that it could be used in future experiments. The design variable *dv_rho* was used under the *modelo.m* MATLAB program to assemble the finite element model with the corresponding density perturbation introduced.

4. Equality Constraints

No equality constraints were used for this experiment, hence its corresponding brackets in the matlab code were left blank.

5. Inequality Constraints

A beam weight inequality constraint was set up in order to bound the weight of each element by a 10%, as well as the total amount of mass that could be added to the entire beam by a 10%.

D. RESULTS AND DISCUSSION

The three Objective Functions were minimized subject to the following constraints:

- System totally unconstrained.
- System with limited amount of mass added to any one element (10 percent).
- System with limited total amount of mass added to the entire beam (10 percent).

The blue bars of the following figures represent the normalized damage detection distribution along the cantilever beam model. The top graphs of each figure represent the analytical damage detection, while the lower graphs represent the experimental damage detection distributions from the laboratory. The stem plots with the yellow circles on the top indicate the magnitude and localization of the actual errors.

1. Modal Assurance Criterion (MAC) Objective Function

Figures 98, 99 and 100 represent the analytical versus experimental comparison of the MAC objective function when it was used for the system totally unconstrained, for a one element mass constrained system and for the total mass constrained system, respectively.

The eigenvector objective function proved to work very consistently, since the maximum value in the table was detected at element 35 where the actual error was located, although there was some error influence at the neighboring elements (34 and 36). The best damage detection/ error prediction performance was found at the total mass constrained condition, as it is shown in Figure 100, where the error is literally minimized and bounded at element 35, with no big influence on the rest of the elements of the cantilever beam.

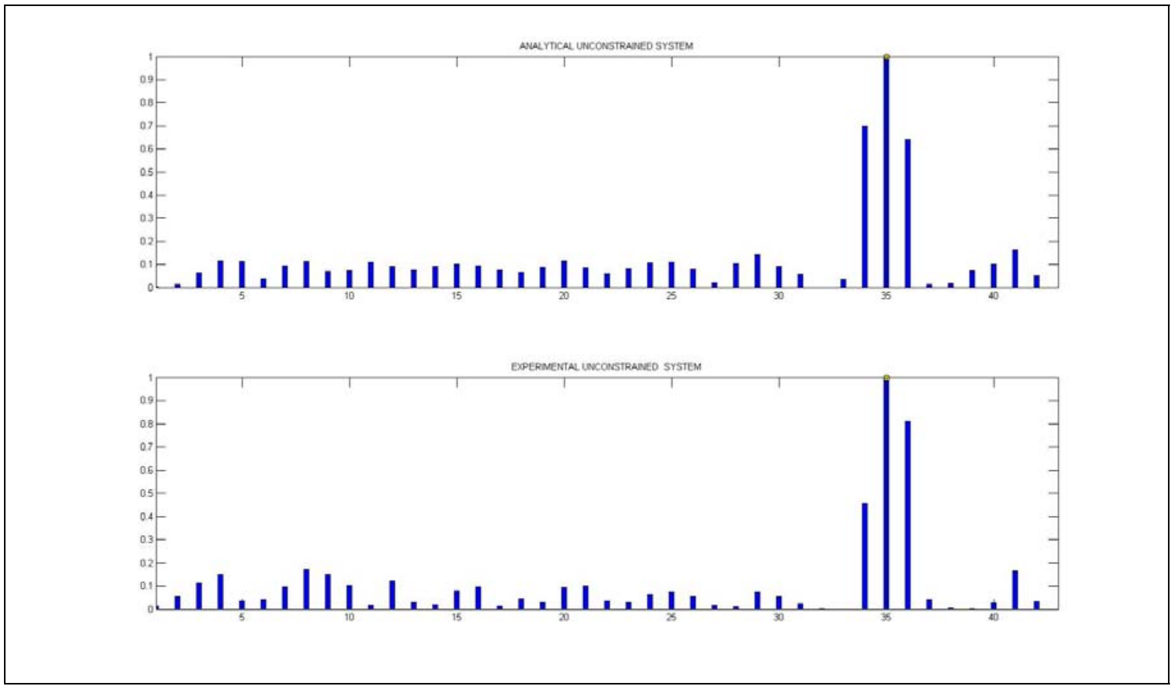

Figure 98. Analytical- Experimental Unconstrained MAC O.F.

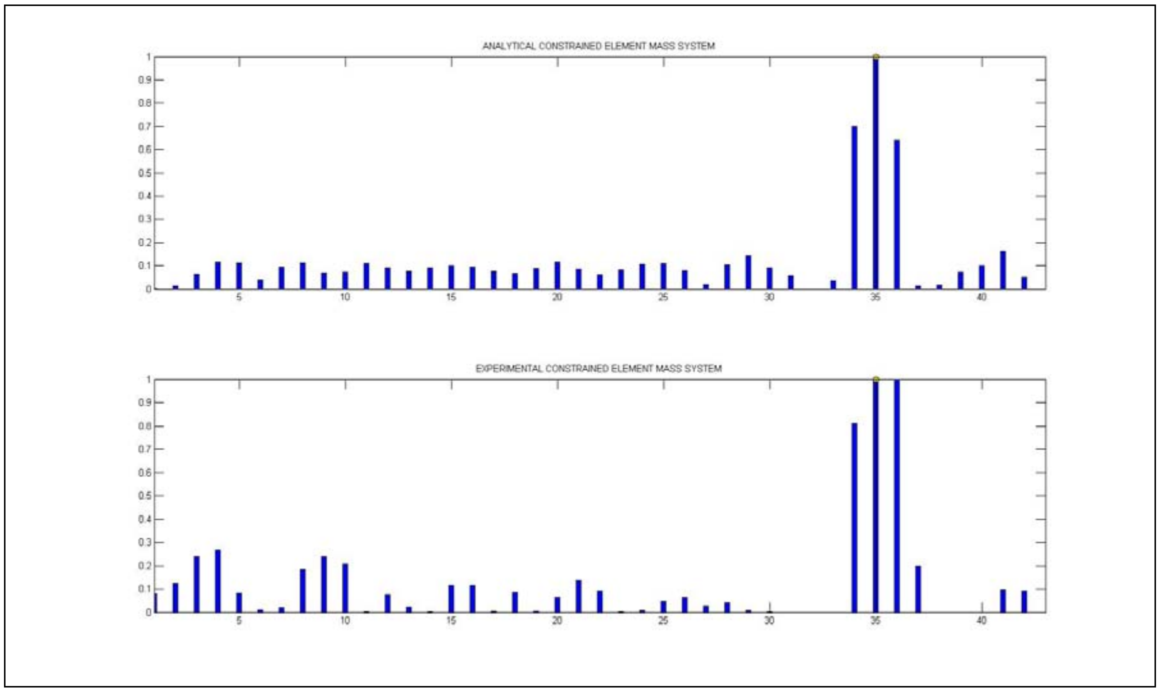

Figure 99. Analytical- Experimental Element Mass MAC O.F.

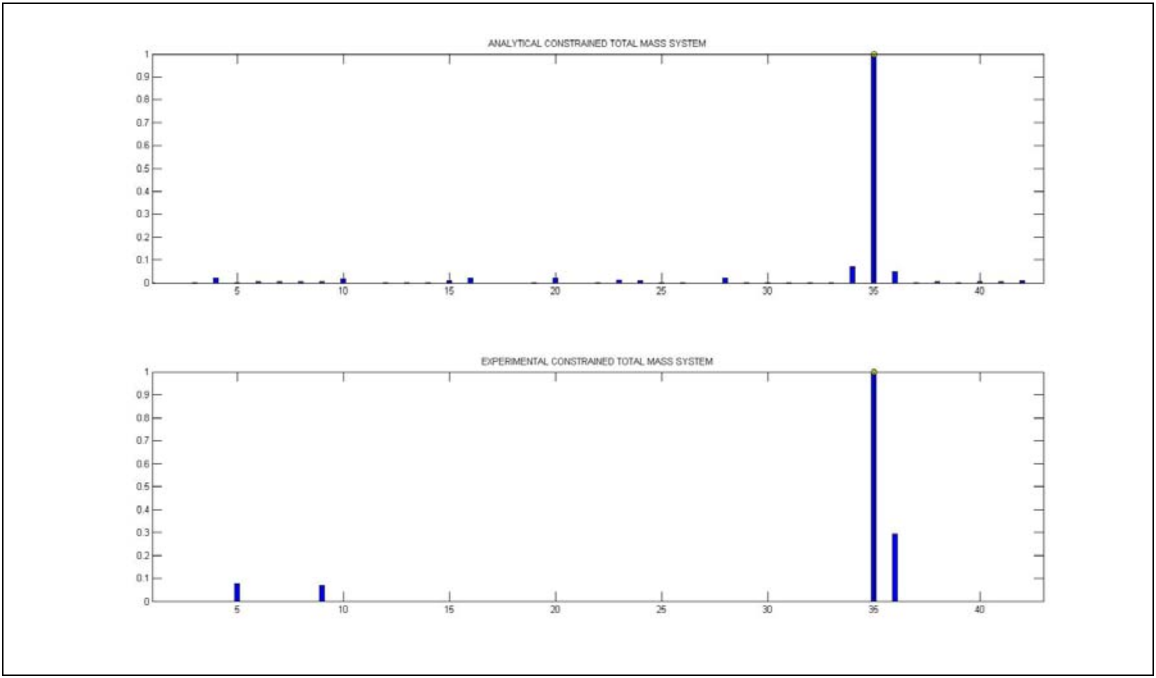

Figure 100.   Analytical- Experimental Total Mass MAC O.F.

2. Combined MAC and Eigenvalue Objective Function

Figures 101, 102, and 103 represent the analytical versus experimental comparison of the MAC- Eigenvalue objective function when it was used for the system totally unconstrained, for a one element mass constrained system and for the total mass constrained system, respectively.

This combined objective function proved to work almost as consistent as the MAC objective function did, yet again, the more constrained the problem became the more isolated the error was bounded as it can be seen in Figure 103. Theoretically speaking the more refined the objective function becomes, the better results are going to be found, in this case this combined objective function did not perform better than the MAC objective function alone, the most likely reason is due to the fact that the Eigenvalue objective function (equation 7.8) is not accurately predicting the damage therefore is not positively contributing to the combined objective function so it can build up a better performance.

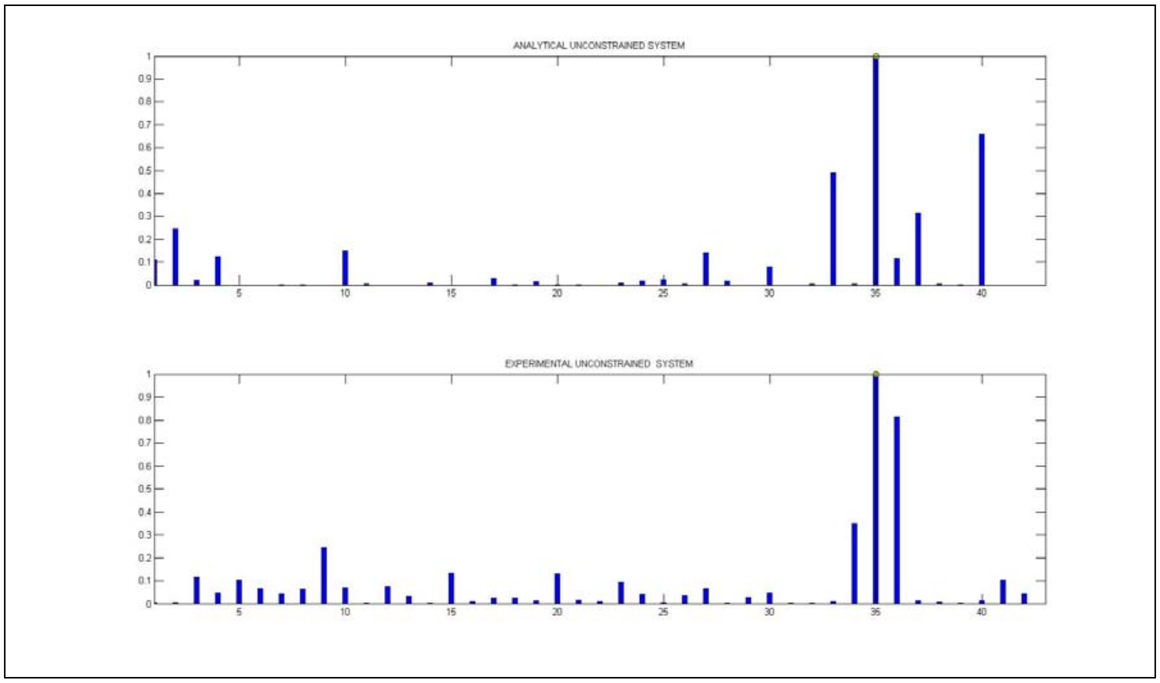

Figure 101. Analytical- Experimental Unconstrained MAC/Eigenvalue O.F.

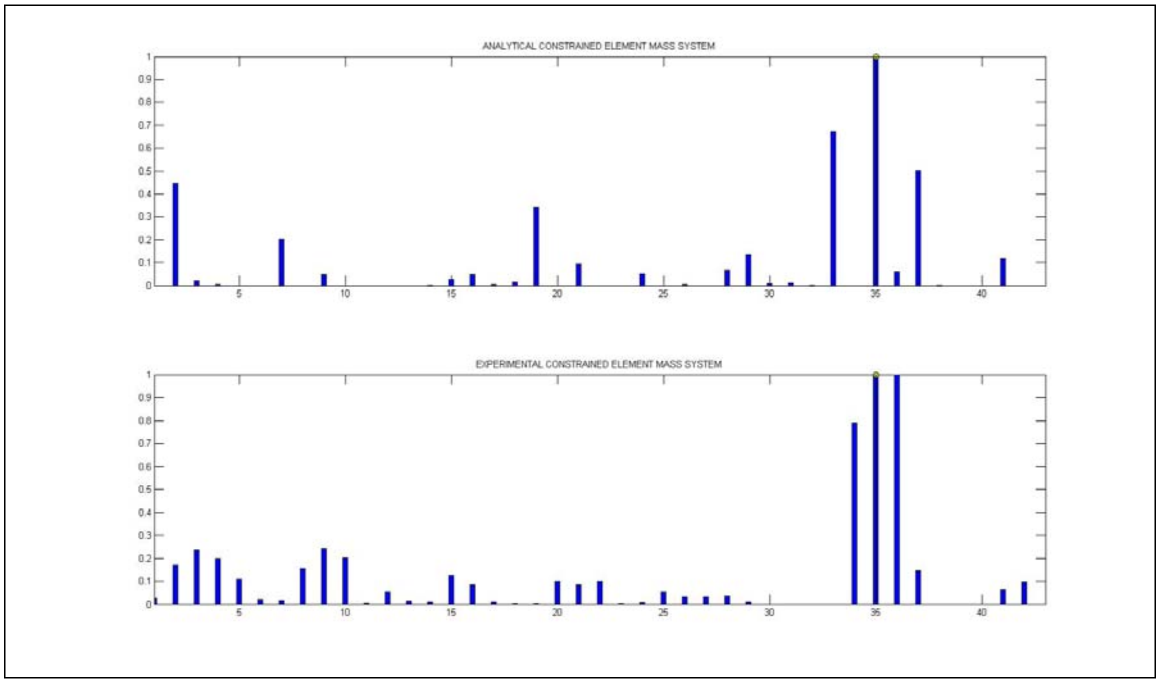

Figure 102. Analytical- Experimental Element Mass MAC/Eigenvalue O.F.

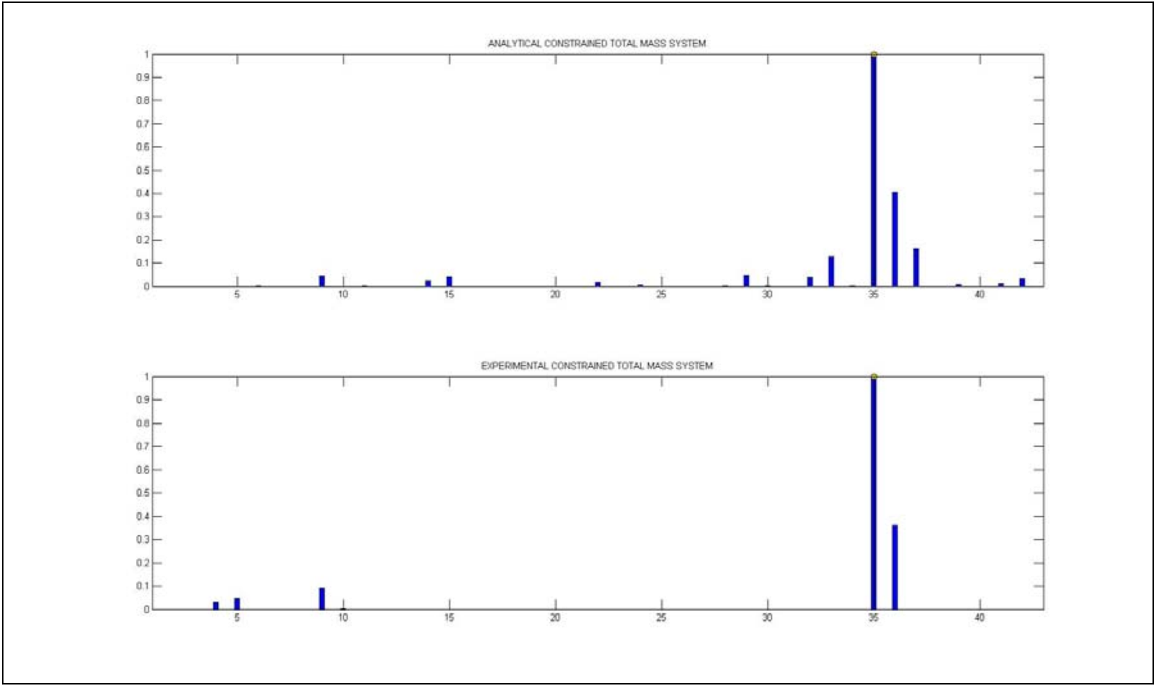

Figure 103. Analytical- Experimental Total Mass MAC/Eigenvalue O.F.

3. Eigenvalue Objective Function

Figures 104, 105, and 106 represent the analytical versus experimental comparison of the Eigenvalue objective function when it was used for the system totally unconstrained, for a one element mass constrained system and for the total mass constrained system, respectively.

The use of the natural frequencies as objective function did not provide a good error estimate and seemed to work as the worst objective function out of the three selected. The main problem was the exceeded number of iterations with the program routine; hence this caused the program to be stopped, before meeting the desired tolerance conditions. The best error prediction was contained within the seventy percent of the actual error, not a very reliable situation.

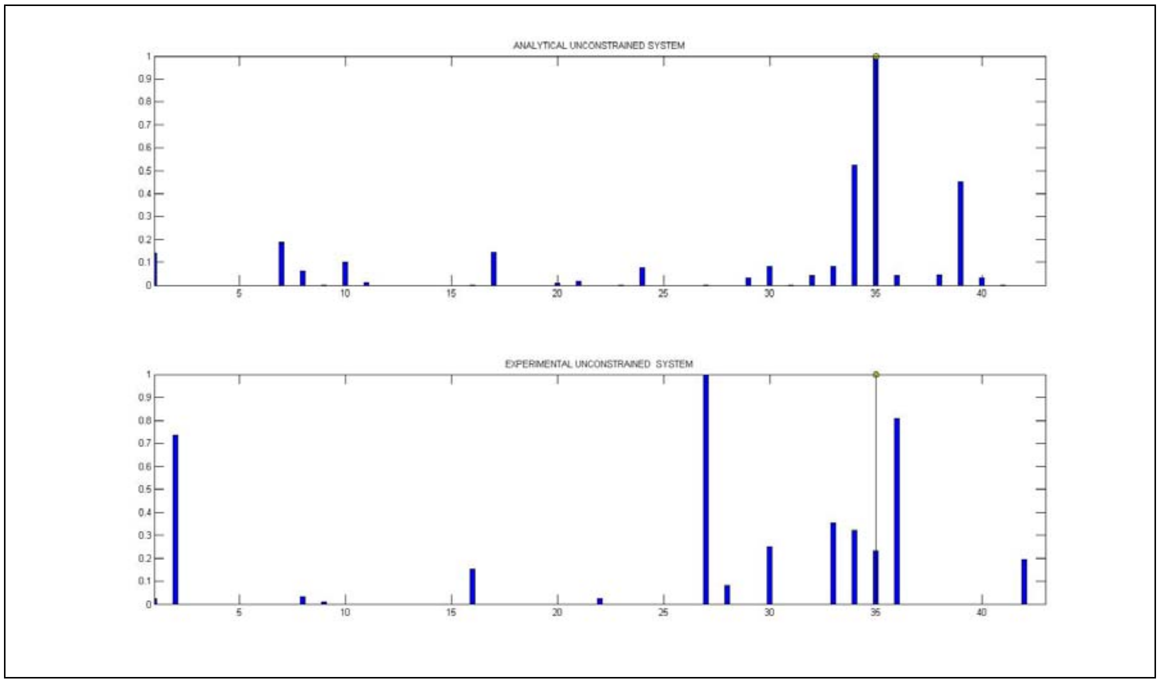

Figure 104. Analytical- Experimental Unconstrained Eigenvalue O.F.

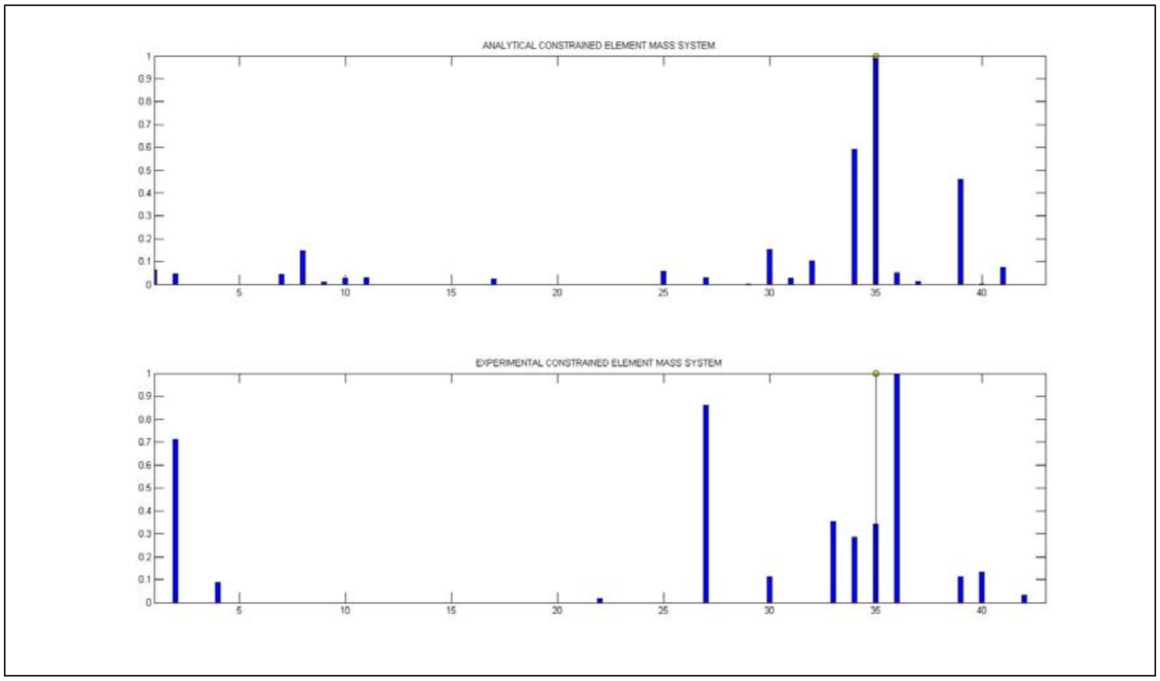

Figure 105. Analytical- Experimental Element Mass Eigenvalue O.F.

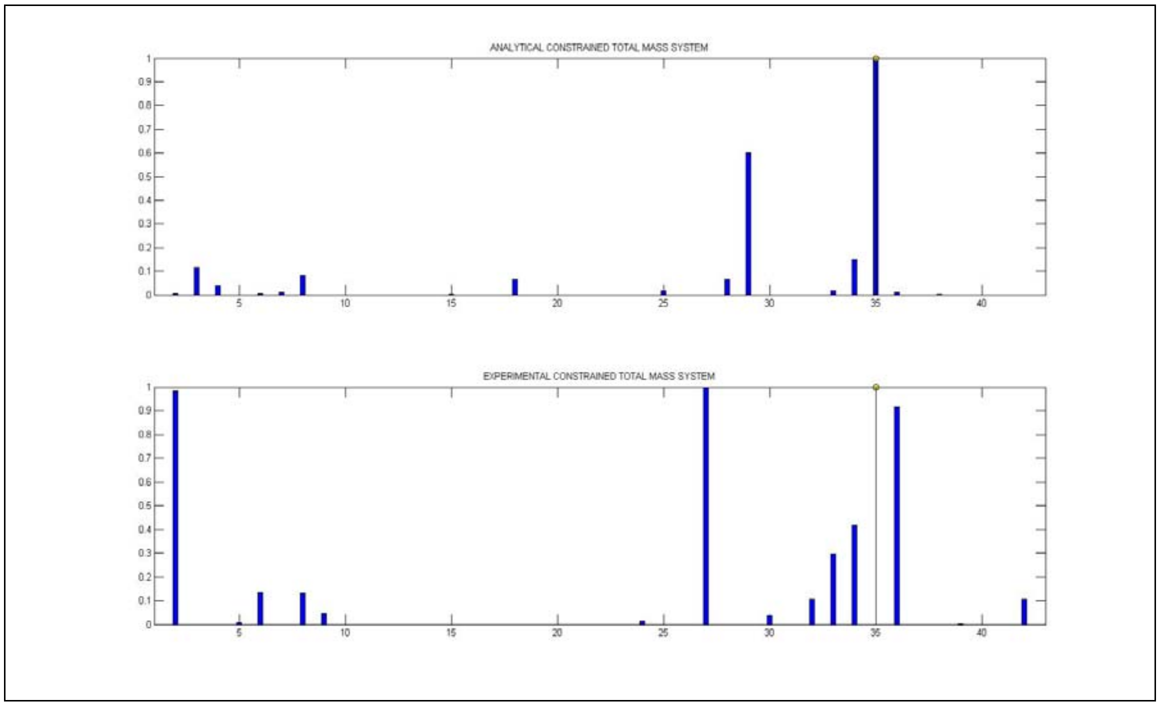

Figure 106. Analytical- Experimental Total Mass Eigenvalue O.F.

The objective function that proved to work the best was the MAC; the combined MAC- Eigenvalue objective function did improve the results from just the natural frequencies objective function, but did not get better results with respect to the MAC objective function alone.

The MAC objective function verified to work more efficiently than the eigenvalues objective function, proving the simplicity and low cost of the natural frequencies ratio objective function, but at the same time it confirms the disadvantages in the damage detection limitation, because the natural frequencies can not provide the spatial information about structural damage. On the other hand, the mode shapes objective function is more sensitive to damage, due to the strain energy correlation at that location (Jaishi and Ren, 2006). Further more the MAC, proved its usefulness in the mode shapes correlation, minimizing the objective function in question.

Theoretically speaking the more refined objective function should give the best damage detection results, but as it was proved with the combined MAC- Eigenvalue objective function, the reality not always matches with the theory, because it is not only a matter of having a more complete/ refined objective function, but it is also required to assure that the chosen parameters assure reliable results by themselves in order to improve the results when combined with other parameters and not to weaken the actual performance of the more complex/ refined objective function. Even when we had the best computer program, if the objective function is not refined enough, the results will not be satisfactorily found.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS AND RECOMMENDATIONS

Artificial Boundary Conditions have been verified to work as a method to find additional natural frequencies by using only one experimental data base, as opposed to reconfiguring the complete experiment such that more than the baseline frequencies could be found. Furthermore, the main purpose of this thesis was to accomplish the damage detection/ error prediction on sensitivity based finite element models by the judicious use of the Artificial Boundary Conditions.

A. CONCLUSIONS

1. The use of Artificial Boundary Conditions via the reduced order model proved to be an effective method to solve for the spatially incomplete real world structures. Hence reducing the disadvantages of the underdetermined problems.
2. The sensitivity based updating governing equation proved to be a very useful tool in localizing the error/ damage, when the Artificial Boundary Conditions frequencies were added to the baseline frequencies.
3. The stiffness sensitivity distribution was directly correlated to the strain energy distribution of the system; consequently the stiffness sensitivity is a function of the flexural properties of the system structure.
4. The mass sensitivity distribution was directly correlated to the kinetic energy distribution of the system; as a result the mass sensitivity is a function of the inertia of the system when an excitation frequency is imposed over the system structure.
5. Damage/ errors were very likely to be detected at areas of high sensitivity distribution of the structure.

6. The damage detection/ error prediction was consistently improved when the ABC's were selected in a manner that increased sensitivity in the desired region.

7. Stiffness errors proved a very consistent behavior. The highest damage detection probability was when the stiffness was the closest to the fixed end of the experimental cantilever beam. The use of the pinned ABC at a DOF adjacent to the element error, consistently improved the damage detection possibilities. When the pinned ABC was applied to a DOF further away with respect to the location of the damaged element, the error was less likely to be detected. The use of multiple ABC's considerably improved the error detection capability of the FEM problem.

8. Mass errors did not provide consistent evidence of damage detection. The only conclusion found was that since the mass errors are related to the kinetic energy of the system, the pinned ABC needs to be set up such that it is able to produce a considerable displacement at the location of the damaged element. Pinned ABC's applied to mass based sensitivity systems tends to diverge the sensitivity, hence any ABC applied at a DOF adjacent to the mass damaged element, is less likely to find the error perturbation. The use of multiple ABC's did not show an improvement as far as the error detection capability is concerned.

9. The optimization program considerably improved the mass error detection performance of the cantilever beam model. The more refined the Objective Function, the more accurate/ minimized results are produced.

10. The modal to spatial transformation in the Optimization routine was accomplished by the use of the "Simple Method", although this method proved to work, it is believed that some more accurate methods would give a more accurate data transfer/conversion, as far as the softwares interface between the MEscopeVES – MATLAB is concerned.

B. RECOMMENDATIONS

1. Find a method of approach to reliably predict the damaged mass elements over the beam model.
2. Find or develop some interface routine to successfully pass directly the data generated between the software used for the finite element model simulation, such MEscopeVES, and the software used for the optimization computer program (MATLAB).
3. Try more than only one physical parameter of the structure to be used as design variables in the formulation of the Objective Function, to allow isolation of all modeling errors while not requiring excessive iterations, hence obtaining an improved optimization result.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

A. ABCRUNTHRU_RLA

```
%  ********************  ABCrunTHRU_rla.m  ***********************

% This program calculates the condition number of the following
% sensitivity matrices used to calculate the DV (error prediction).
% 1) Base system only 5 modes (underdetermined)
% 2) ABC system 10 modes
% 3) Base system 5 modes + 5 modes from ABC system
% The last system is calculated 3 times.  Once for modes 1-5,
% another for modes 6-10, and again for modes 11-15.
%
% This program is called from Build2Beams.m.
%
% Written by Constance R S Fernandez, Spring 2004
% Updated by John R Mentzer, Spring 2007

%INPUTS
% --------
% icnt_oset
% T_sens_tot
% vect_lam_tot

% OUTPUTS
% -------
% aa
% dv_a
% dv_b
% dv_c
% intervel
% mode
% startmode
% ten
% dv_cal_ABC - matrix
% dv_calABCten - matrix
% dv_cal_BasePlus - matrix
% cond_ABC - matrix
% cond_ABCten - matrix
% cond_basePlus - matrix

% ----INITIALIZATION-------

abc = 0;
ten = 1;
intervel = 1;
int_abc = 1;
for aa = 1:icnt_oset +1 % number of conditions (base + ABC)
    a_c = 1; % reinitialize for each ABC system

    for mode = 1:3 % 3 sets of modes per ABC system (10 element beam)
        % (modes 1:5, 6:10, 11:15)
        startmode = abc + a_c;

        dv_a = [startmode: startmode+4];    % modes
        dv_d1 = [ten:ten];     % mode 1
        dv_d2 = [ten+1:ten+1];   % mode 2
        dv_d3 = [ten+2:ten+2];   % mode 3
        dv_d4 = [ten+3:ten+3];   % mode 4
        dv_d5 = [ten+4:ten+4];   % mode 5


        dv_c = [ten:ten+9]; % the first 10 modes of each ABC system
        % (modes 1-10 only)

        if dv_a == [1:.25*size(T_sens_tot,2)]; % if sensitivity matrix
            %has only 5 rows than the modes used are all 5, else modes
            %used are first five (base) and a set of selected 5 modes
            %of ABC system for a total of 10 modes.
            dv_b = [dv_a];
        else
```

```
            dv_b = [1:.25*size(T_sens_tot,2), dv_a];
        end
        %---Base System only---(underdetermined)

        % save DV calculates of as matrix for plotting
        dv_cal_ABC(:,intervel) = T_sens_tot(dv_a,:)\vect_lam_tot(dv_a); hold on
        % condition number of Sensitivity matrix used in DV cal.
        cond_ABC(intervel,1) = cond(T_sens_tot(dv_a,:));
       % cond_ABC(intervel,1) =
cond((T_sens_tot(dv_a,:))*(T_sens_tot(dv_a,:))');%(Ttrans)(T)
%
%         %mode 1
%          dv_cal_ABC1 = T_sens_tot(dv_d1,:)\vect_lam_tot(dv_d1);
%         %mode 2
%          dv_cal_ABC2 = T_sens_tot(dv_d2,:)\vect_lam_tot(dv_d2);
%         %mode 3
%          dv_cal_ABC3 = T_sens_tot(dv_d3,:)\vect_lam_tot(dv_d3);
%         %mode 4
%          dv_cal_ABC4 = T_sens_tot(dv_d4,:)\vect_lam_tot(dv_d4);
%         %mode 5
%          dv_cal_ABC5 = T_sens_tot(dv_d5,:)\vect_lam_tot(dv_d5);


        % ---Base + 5 modes of ABC system ----
        dv_cal_BasePlus(:,intervel) = T_sens_tot(dv_b,:)\vect_lam_tot(dv_b);
       cond_basePlus(intervel, 1) = cond(T_sens_tot(dv_b,:));
       % cond_basePlus(intervel, 1) =
cond(((T_sens_tot(dv_b,:))*(T_sens_tot(dv_b,:))'));%(Ttrans)(T)

        intervel = intervel + 1;
        a_c = a_c + 5; % five modes used at a time

    end % "mode" loop

    % ---ABC system only ----



       dv_cal_ABCten(:,int_abc) = T_sens_tot(dv_c,:)\vect_lam_tot(dv_c);
       cond_ABCten(int_abc,1) = cond(T_sens_tot(dv_c,:));
        %cond_ABCten(int_abc,1) =
cond(((T_sens_tot(dv_c,:))*(T_sens_tot(dv_c,:))'));%(Ttrans)(T)

    int_abc = int_abc + 1;
    abc = abc + 39; % advances to next ABC system, must change number "19"
    % to reflect the number of DOF in beam.  This beam had 10 elements thus
    % 19 DOF.
    ten = ten + 39; % advances to next ABC system
end % "aa" loop


% ********************  END ABCrunTHRU_rla.m  ***********************
```

B. ADDLUMPMASS_RLA

```
% ********************  AddLumpMass_rla.m  ***********************

%  This script constructs a vector of lumped masses
%  which is added to the diagonal of the BeamX mass matrix.
%          Mass added to [mx] in Assemble2Beams.m
%
%  Written by Prof J.H. Gordis

% Inputs needed
% -----------------
% num_elements
% mx

% Outputs
% -----------------
% mass_diag
% mass_node
% mass_coord
% mass_DOF
% mx - updated

disp(' ');disp(' ');
disp(' ******************************************************')
disp(' ****         Lumped mass addition to beams        ****')
disp(' ****  Lumpmass DOFs defined for UNRESTRAINED beams  ****')
disp(' ******************************************************')

% initalize
if exist('mass_diag') == 0;  % define and apply lumped mass vector.

add_mass = 'n';
add_mass = input(' Add lumped masses to BeamX ? (y/n) ','s');

% Initialize vector to add to [mx] diagonal.

mass_diag = zeros(2*(num_elements+1),1);

  while add_mass == 'y';

    mass_node = input(' Node number for lumped mass ? ');

    mass_coord = input(' Translation or Rotation for lumped mass ? (t/r) ','s');

    if mass_coord == 't'; % Translational lumped mass
        mass_DOF = 2 * mass_node - 1;
    elseif mass_coord == 'r'; % rotational lumped mass
     mass_DOF = 2 * mass_node;
 end

 mass_diag(mass_DOF) = input(' Enter value of mass/inertia (in "lbf-sec^2/in" ');
    % puts lumped mass on correct DOF
 add_mass = input(' Add another lumped mass ?  (y/n) ','s');
    % can continue adding mass until 'n' is entered

  end;   % End while loop

end;   % End exist('mass_diag')

mx = mx + diag(mass_diag);   % Add lumped masses to [mx]:

% ********************  END ADDLUMPMASS_rla.m  **********************
```

C. ASSEMBLESENS_RLA

```
% ********************  AssembleSens_rla.m  ***********************
%
% This program assembles the total sensitivity matrix, T_sens_tot and
% total lam vector,  vect_lam_tot and assembles the relative frequency
% error between the natural frequencies of Beam A and Beam X
% Written by Constance R S Fernandez, Spring 2004
% Updated by John R Mentzer, Spring 2007



% INPUTS
% -------
% vect_lamx_oset
% vect_lam_oset
% vect_lam
% T_sens_oset
% T_sens
%
% OUTPUTS
% -------
% vect_OSET
% vect_lam_tot
% T_sens_tot
% freq_OSET
% freq_OSETx
% rel_freqERROR

vect_OSET = vect_lamx_oset - vect_lam_oset;
% lamx from actual beam with error oset, lam from FE model oset
% Creating a vector  of lam differences calculated (Lx-La)
if vect_OSET == 0;
    % when vector is empty at first, the total vector is equal to the
    % lam vector of Beam A, i.e., the first 19 values of vect_lam_tot are
    % the natural freq squared (rad^2/sec^2) of Beam A

    vect_lam_tot = vect_lam;
else
      vect_lam_tot = cat(1, vect_lam, vect_OSET);
end



if T_sens_oset == 0;

    T_sens_tot = T_sens;
else
    T_sens_tot = cat(1, T_sens, T_sens_oset);
end

freq_OSET = sqrt(abs(vect_OSET))/2/pi;
% Natural frequency vector of Beam A in Hz
freq_OSETx = sqrt(abs(vect_lamx_oset))/2/pi;
% Natural frequency vector of Beam X in Hz
rel_freqERROR = freq_OSET./freq_OSETx*100;
% Relative error between Beam A OSET natural freq and Beam X OSET natural Freq.

% *******************  END AssembleSens_rla.m  **********************
```

D. BEAMPROPERTIES_RLA

```
% ********************  BeamProperties_rla.m  ***********************

% This is the "props_file" to load nominal beam data.
% This program is called by BeamA_Prompt_crs to provide beam properties
% in order to build Beam A.

% This program was written by Constance Fernandez, Spring 2004
% This progam modified by John R Mentzer, Spring 2007
% Outputs
% -------
% depth
% width
% E
% rho
% total_length
% num_elements
% nominal_EI
% nominal_area
% nominal_density

% Following are actual measurements from experimental set-up cantilever
% beam.
 depth = .504;% in z-dir (inches)
 width = 1.506; % in y-dir (inches)
 E = 10e6;
    %E = 1.65e6; % lbf/sec^2-in (10e6-ksi)
    %(1bf/in^2 = 6894.76Pa)-> E(lbf/in^2) = ()Pa/6894.76
 rho =0.110460934; %0.098;% lbf/in^3

    % T6 temper alloys require a 35-ksi tensile strength, 30-ksi yield
    % strength and a 10e6-ksi elastic modulus. Alloy 6061-T6 has 1.0
    % pct magnesium, 0.6 pct silicon, 0.3 pct copper and 0.2 pct chromium.
    % It has a 45-ksi tensile strength and 35-ksi yield strength.1 The
    % machinability of aluminum alloys are high (300) compared to titanium
    % (40). Aluminum alloys can easily be bent and provide easy loading and
    % unloading of parts. Also, aluminum is a highly conductive metal
    % compared to titanium.

% all measurement of distance are in inches
 total_length    = 42;
 num_elements    = 20;
 nominal_EI      = (width * depth^3 / 12) * E;
 nominal_area    = depth * width;% in^2
 nominal_density = rho;% lbf/in^3

% *******************  END BeamProperties_rla.m  **********************
```

E. BEAMSENSITIVITY_RLA

```
% ********************  BeamSensitivity_rla.m  ***********************

% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 4/4/95  Basic frequency sensitivities
%  Updated: Spring 2004  Constance R S Fernandez

% *******************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~
%
%  This program calculates mode frequency sensitivities as given by the
%  equation,
%
%         ¶w^2               ¶[k]        ¶[m]
%         --    =  {P}' * [  ---- - w^2 --- ] * {P}
%         ¶DV                ¶DV         ¶DV
%
%     where:  w  = natural frequency
%            {P} = associated mode shape
%             DV = design variable
%
%  The right side of the above equation is considered the addition of
%  the sensitivity matrix wrt EI and the sensitivity matrix wrt mass.
%
%  The program calculates the stiffness and mass matrix partials by finite
%  differences. That is, for example, the [k] matrix is assembled twice,
%  once in for the nominal beam parameters, and a second matrix is
%  assembled based on a small perturbation of element mass and/or EI.
%
%  This program makes use of the beam data created by the program
%  "Build2Beams.m."
%  1) Resets BeamX mass and EI data to be the same as BeamA data.
%  2) Enters a loop to create a sensitivity matrix (T)
%     In loop
%     a)  A small perturbation of mass is added to BeamX on ele.1.
%     b)  The mass matrix is assembled for this mass-perturbed beam,
%         and the mass matrix partial derivative is calculated as
%
%                 ¶[m]        [m_perturb] - [m_baseline]
%                 ---    =    ------------------------
%                 ¶DV                    ¶DV
%
%     c)  The first column of Sensitivity matrix is calculated using:
%
%            sens_mass = [phi_base]'*(-lam_base)*m_delta*[phi_base]
%
%     (Note: A column of T corresponds to the respective element on beam.)
%
%     d) T loop starts again but with the small change on element 2.
%     e) Difference calculated and second column of T is calculated.
%
%   3) loop continues until all columns of T are calculated, corresponding
%     to a small change added to the respective element.
%
%   4) The procedure is identical for stiffness sensitivity.
%      Except:
%      sens_EI = [phi_base]'*k_delta*[phi_base]
%
%   5)Combine the two T's into one complete sensitivity matrix :
%          T_sens = [sens_mass,sens_EI].
%     In words, the first set of columns (equal to number of elements)of
%     the combined matrix is the sensitivity mass wrt mass changes and
%     the last half is wrt EI changes.
%
%
% ***********************************************************************

% Inputs Needed:
% -------------
```

```
% mass_lbls
% EI_lbls
% element_mass
% element_EI
% num_rbm
% num_elements
% lamx (experimently measured nat freq of beam)

% Programs called
% -------------
% Assemble2Beams_crs;
% BoundaryConditions_crs;
% fmodes
% fOset_from_Aset

% Outputs:
% -------------
% num_modes
% lam_base
% phi_base
% sens_mass
% sens_EI
% T_sens
% dv_cal
% freq_base
% vect_lam

%  Start code:
%  ~~~~~ ~~~~~
format long;


% *******************************************************************
% ************************ INITIALIZATION **************************
% *******************************************************************

mass_change = 1; % Percent mass change for sensitivity calculation.
EI_change = 1;   % Percent EI change for sensitivity calculation.

element_mass_orig =  element_mass;  % Copy properties to retain them.
element_EI_orig   =  element_EI;

% Prompt for number of mode frequencies to generate sensitivities for:
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%num_modes = input(' Enter number of modes for sensitivity calculations>> ');
num_modes = 39; % 39 is maximum number of modes in a 20 element cantilever
                % beam.  This is hard coded for quicker run of simulation.

start_mode = num_rbm + 1;   % Skip the rigid body modes.

% *******************************************************************
% ************* MASS SENSITIVITY CALCULATION LOOP *******************
% *******************************************************************

sens_mass = 0; % initialize Mass based sensitivity matrix

if mass_lbls ~= 0; % From Beam_XPrompt as user inputs

    for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

        %   Resetting BeamX properties to BeamA properties
        element_mass(:,2)  =  element_mass(:,1);

        %   each element, one at a time will have a change in mass
        element_mass(icnt_dv,2) = element_mass(icnt_dv,2) * ...
            (1 + mass_change/100);

        Assemble2Beams_crs;   % Run script to assemble beams.

        BoundaryConditions_crs;  % Apply boundary conditions.

        [lam_base, phi_base] = fModes(ka,ma);

        % ma is the basebeam without changes,
        % mx is beam with small change in mass added (mass_change)

        %    Form mass derivative matrices:
        m_delta = (mx - ma)/(mass_change/100);
```

```
        % converts precent change into decimal amount


        %    Mode freq sens loop:
        end_mode = start_mode + (num_modes - 1);
        row_num = 0; % initialize loop
        for icnt_modes = start_mode:end_mode;
            row_num = row_num +1;
            sens_mass(row_num,icnt_dv) = phi_base(:,icnt_modes)' *...
                (-lam_base(icnt_modes) * m_delta ) *...
                phi_base(:,icnt_modes);
            % definition of mass sensitivity matrix
        end; % for "icnt_modes" inner loop

    end;  % End "for icnt_dv" outer loop for sensitivity calculations

end;    % End "if mass_lbls ~= 0"


% *******************************************************************
% ************** EI SENSITIVITY CALCULATION LOOP *********************
% *******************************************************************

sens_EI = 0;  % initialize EI based sensitivity matrix
if EI_lbls ~= 0; % From Beam_XPrompt as user inputs

    for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

        %   Resetting BeamX properties to BeamA properties
        element_EI(:,2)  =  element_EI(:,1);

        %   each element, one at a time will have a change in EI
        element_EI(icnt_dv,2) = element_EI(icnt_dv,2) * ...
            (1 + (EI_change/100) );

        Assemble2Beams_rla;   % Run script to assemble beams.

        BoundaryConditions_rla;  % Apply boundary conditions.

        [lam_base, phi_base] = fModes(ka,ma);


        % ka is the basebeam without changes,
        % kx is beam with small sensitivity added

        %    Form EI derivative matrices:
        k_delta = (kx - ka)/(EI_change/100);
        % converts precent change to decimal value

        %    Mode freq sens loop:
        end_mode = start_mode + (num_modes - 1);
        row_num = 0;

        for icnt_modes = start_mode:end_mode;
            row_num = row_num +1;
            sens_EI(row_num,icnt_dv) = phi_base(:,icnt_modes)' *...
                k_delta * phi_base(:,icnt_modes);
            %definition of EI sensitivity matrix
        end; % for "icnt_modes" inner loop

    end; % End "for icnt_dv" outer loop for sensitivity calculations

end;    % if EI_lbls = 0;


% Copy element EI and mass properties back into arrays:
element_EI   = element_EI_orig;
element_mass = element_mass_orig;

% cleans up workspace by clears all unimportant parameters
clear element_EI_orig element_mass_orig end_mode start_mode
clear row_num icnt_modes EI_change k_delta mass_change m_delta
clear ka kx ma mx icnt_dv

% Assembles complete sensitivity matrix
if sens_mass == 0& sens_EI ~=0;

    T_sens = sens_EI; % resultant sensitivity matrix is equal
    % to EI sensitivity matrix with only EI changes if no inputs
    % are given for mass changes

elseif sens_mass ~= 0 & sens_EI == 0;
    T_sens = sens_mass; % resultant sensitivity matrix is equal
    % to Mass sensitivity matrix with only mass changes if no inputs
    % are given for EI changes

else
    T_sens = cat(2, sens_mass,sens_EI);% else the resultant sensitivity
    %matrix is teh combination of mass sensitivity matrix in first set
    %of columns and EI sensitivity matrix in the last set of columns

end

freqx = sqrt(lamx)/2/pi;
%freq_base = sqrt(lam_base)/2/pi;
% NOTE: % lamx = experiment measured natural freq of beam
vect_lam = (lamx(1:num_modes)-lam_base(1:num_modes));

% ********************  END BeamSensitivity_rla.m  ***********************
```

F. BEAMSENSITIVITYOSET_RLA

```
% ********************  BeamSensitivityOSET_rla.m  **********************

% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 4/4/95  Basic frequency sensitivities
%  Updated: Spring 2004  Constance R S Fernandez to create resulting
%  sensitivity matrix using lam vector of multiple ABC systems

% *********************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~~
%
%  This program calculates mode frequency sensitivities as given by the
%  equation,
%
%         ¶w^2                ¶[k]       ¶[m]
%         --    =  {P}' * [  ---- - w^2 --- ] * {P}
%         ¶DV                 ¶DV         ¶DV
%
%    where:  w  = natural frequency
%           {P} = associated mode shape
%            DV = design variable
%
%  The right side of the above equation is considered the addition of
%  the sensitivity matrix wrt mass and/or EI.
%
%  The program calculates the stiffness and mass matrix partials by finite
%  differences. That is, for example, the [k] matrix is assembled twice,
%  once in for the nominal beam parameters, and a second matrix is
%  assembled based on a small perturbation of element mass and/or EI.
%
%  This program makes use of the beam data created by the program
%  "Build2Beams.m."
%  1) Resets BeamX mass and EI data to be the same as BeamA data.
%  2) Enters a loop to create a sensitivity matrix (T)
%     In loop
%     a)  A small perturbation of mass is added to BeamX on ele.1.
%     b)  The mass matrix is assembled for this mass-perturbed beam,
%         and the matrix partial derivative is calculated as
%
%                 ¶[m]        [m_perturb] - [m_baseline]
%                 ---    =    ------------------------
%                 ¶DV                   ¶DV
%
%     c)  The first column of Sensitivity matrix is calculated using:
%
%            sens_mass = [phi_base]'*(-lam_base)*m_delta*[phi_base]
%
%     (Note: A column of T corresponds to the respective element on beam.)
%
%     d) T loop starts again but with the small change on element 2.
%     e) Difference calculated and second column of T is calculated.
%
%   3) loop continues until all columns of T are calculated, corresponding
%     to a small change added to the respective element.
%
%   4) The procedure is identical for stiffness sensitivity.
%      Except:
%      sens_EI = [phi_base]'*k_delta*[phi_base]
%
%   5)Combine the two T's into one complete sensitivity matrix :
%          T_sens = [sens_mass,sens_EI].
%     In other words, the first set of columns (equal to number of elements)of
%     the combined matrix is the sensitivity with respect to (wrt) mass
%     changes and the last half is wrt EI changes.
%
%
% ****************************************************************************

% Inputs Needed:
% --------------
% mass_lbls
% EI_lbls
% element_mass
```

```
% element_EI
% num_rbm
% mx_beam
% kx_beam

% Programs called
% --------------
% Assemble2Beams_crs;
% BoundaryConditions_crs;
% fmodes
% fOset_from_Aset
% displacmentPlot_OSET

% Outputs:
% --------------
% num_modesO
% sens_massO, sens_EIO
% num_rbmOSET
% T_sens_oset
% dispX_tot, dispA_tot
% accel_plot
% oset_choice
% accelometer
% BC, BCose, BCOSET
% remaindof
% ASETtot, OSET, OSETtot
% inct_sens
% mass_change, EI_change
% phiXPLOT, phiAPLOT
% maO_base, ka0_base
% mxO, kxO
% plotmx, plotkx
% lamaOSET, phiaOSET
% lamxOSET, phizOSET
% lamxplot, phixplot
% faO
% m_deltaO, k_deltaO

%  Start code:
%  ~~~~~ ~~~~~

% ********************************************************************
% ************************* INITIALIZATION ***************************
% ********************************************************************
T_sens_oset = [];
vect_lam_oset = [];
dispX_tot  = [];
dispA_tot  = [];
inct_sens = 0;
icnt_oset = 0;
BCOSET = zeros(ndof, ndof);
OSETtot = zeros(ndof, ndof);
ASETtot = zeros(ndof, ndof);

oset_choice = 'n';
ndof % prints ndof to screen for user's reference

accelometer = [3:2:ndof]; %This needs to change if you do not use a clamped free
beam!
%accelometer = [3 5 7 9 11 13 15 17 19 21]; % use this line for convenience
% in quicker calculation loops or use the next 2 lines.
%accelometer = input('On what nodes are the accel. located','s'); %requests
% user to input locations of accelometers
%accelometer = eval(['[',accelometer,']']); % converts string to vector of
% labels

% saved for plotting accelometers in correct position.
accel_plot = floor(accelometer/2)+1;
% graphical representation of accelometer locations

oset_choice = input('  Do you want to use ASET and OSET (y/n)? ','s');
% user can choose to use ASET and OSET calculations
while oset_choice ~= 'n';
    icnt_oset = icnt_oset +1;
    disp(' locations of Accel.') % displays "location of Accel" on screen
    accelometer % displays the vector of location of Accel for user's
    % reference in choosing which DOF are to be pinned
    disp(' ')
    disp('   Enter pinned DOF label(s)')
```

```
    disp('   Use MATLAB vector format> 1 3 5:7 9  ')
    pinned = input('   >> ','s');
    pinned = eval(['[',pinned,']']); % Converts string to vector of labels
    BC(icnt_oset,:) = [1,2,pinned]; % Boundry conditions for cantilever beam,
    % change line if different beam is used

    BCoset = fOset_from_Aset(ndof, BC(icnt_oset,:)); % gets OSET from ASET

    BCOSET(icnt_oset, 1:length(BCoset)) = BCoset; % copies OSET into another
    % vector to be used

    % loop to find remaining accelometers.
    for icnt = 1 : length(pinned);
        remain(icnt,:) = find(pinned(icnt) == accelometer);
    end

    remaindof = fOset_from_Aset((length(accelometer)), remain);
    % remaining unpinned DOF

    % remaining accelometers
    aset = accelometer(remaindof);
    ASETtot(icnt_oset, 1:length(aset)) = aset;

    % omitted oset, i.e. pinned accelometers
    OSET = fOset_from_Aset(ndof, aset);
    OSETtot(icnt_oset, 1:length(OSET)) = OSET; % copies OSET into another
    % vector to be used in======

    inct_sens = inct_sens+1;

    mass_change = 1;  % Percentage mass change for sensitivity calculation.
    EI_change = 1;    % Percentage EI change for sensitivity calculation.

    element_mass_orig =  element_mass;  % Copy properties over to retain them.
    element_EI_orig   =  element_EI;    %


    % Prompt for number of mode frequencies for which to generate sensitivities
    % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % use the following three lines for user's input for number of modal
    % frequencies for which to generate sensitivity or use the fourth line
    % which uses the maximum number of modes for the defined system

    %disp ('max number of modes  ')
    %length(oset)
    %num_modesO = input('  Enter number of elastic modes for sensitivity
calculations>> ');
    num_modesO = ndof - length(BC(icnt_oset,:));%max number of modes in system
    start_mode = num_rbm + 1;   % Skip the rigid body modes.

    % Initializes two vectors to be used in ploting program.
    phiXPLOT = zeros(ndof,num_modesO); %cantilever beam
    phiAPLOT = zeros(ndof,num_modesO); %cantilever beam


    % *******************************************************************
    % ************* MASS SENSITIVITY CALCULATION LOOP ******************
    % *******************************************************************

    sens_massO = 0;
    if mass_lbls ~= 0; % from Beam_XPrompt as user inputs

        for icnt_dv =  1:num_elements;  % loop to create sensitivity matrix

            %   Resetting BeamX properties to BeamA properties.
            element_mass(:,2)  =  element_mass(:,1);

            %   each element, one at a time will have a change in mass
            element_mass(icnt_dv,2) = element_mass(icnt_dv,2) * ...
                (1 + mass_change/100);

            Assemble2Beams_crs;   % Run script to assemble beams.

            % Articial Boundry Conditions
            maO_base = ma(BCoset, BCoset);
            % new mass matrix of the system defined by OSET
            mxO = mx(BCoset, BCoset);

            plotmx = mx_beam(BCoset, BCoset);
```

```
            % resulting mass matrix with ABC used in plotting
            plotkx = kx_beam(BCoset, BCoset);
            % resulting stiffness matrix with ABC used in plotting

            kaO_base = ka(BCoset, BCoset);
            kxO = kx(BCoset, BCoset);

            % lam (natural freq^2, rad^2/sec^2), phi (mode shapes)
            [lamaOSET,phiaOSET] = fModes(kaO_base,maO_base);
            % natural freq of new artifically bounded base system
            [lamxOSET,phixOSET] = fModes(kxO,mxO);
            % natural freq/ modes of new artifically bounded system with
            % either EI or mass changes added similiar to orginial calculations
            [lamxplot,phixplot] = fModes(plotkx,plotmx);
            %resulting lam & phi of ABC system used in plotting

            % Mode shapes
            if length(pinned)==1 & pinned == 3 % the next DOF acts a little
different
                % because of the location on beam thus it was easier
                % to program it seperately
                phiAPLOT(2:ndof-2, :) = phiaOSET(1:ndof-3, :);
                phiXPLOT(2:ndof-2, :) = phixplot(1:ndof-3, :);

            elseif length(pinned)>1 & pinned(1,1)==3
                phiAPLOT(2:pinned(1,2)-2,:) = phiaOSET(1:pinned(1,2)-3,:)
                phiAPLOT(pinned(1,2)-1:ndof-3,:) = phiaOSET(pinned(1,2)-2:ndof-4,:)


            elseif length(pinned)<2
                phiAPLOT(1:pinned-3, :) = phiaOSET(1:pinned-3, :);
                phiAPLOT(pinned-1:ndof-2,:) = phiaOSET(pinned-2:ndof-3,:);
                phiXPLOT(1:pinned-3, :) = phixplot(1:pinned-3, :);
                phiXPLOT(pinned-1:ndof-2,:) = phixplot(pinned-2:ndof-3,:);



            else
                phiAPLOT(1:pinned(1,1)-3, :) = phiaOSET(1:pinned(1,1)-3, :);
                phiAPLOT(pinned(1,1)-1:pinned(1,2)-3,:) = phiaOSET(pinned(1,1)-
1:pinned(1,2)-3,:);
                phiAPLOT(pinned(1,2)-1:ndof-3,:) = phiaOSET(pinned(1,2)-3:ndof-
4,:);
                phiXPLOT(1:pinned(1,1)-3, :) = phixplot(1:pinned(1,1)-3, :);
                phiXPLOT(pinned(1,1)-1:pinned(1,2)-3,:) = phixplot(pinned(1,1)-
1:pinned(1,2)-3,:);
                phiXPLOT(pinned(1,2)-1:ndof-3,:) = phixplot(pinned(1,2)-3:ndof-
4,:);
           end %




            num_rbmOSET = length(find(lamaOSET < 1));
            % find number of rigid bodies in new ABC system
            start_mode = num_rbmOSET + 1;   % Skip the rigid body modes.

            faO = sqrt(lamaOSET)/(2*pi); %natural freq of the ABC in Hz

            %    Form mass derivative matrices:
            m_deltaO = (mxO - maO_base)/(mass_change/100);
            % converts percent change to decimal value
            % NOTE: mass_change needs to be the same change used in
            % orginial mass sensitivity matrix calculation

            %    Mode freq sens loop:
            end_mode = start_mode + (num_modesO - 1);
            row_num = 0;

            for icnt_modes = start_mode:end_mode;
                row_num = row_num +1;
                sens_massO(row_num,icnt_dv) = phiaOSET(:,icnt_modes)' *...
                    (-lamaOSET(icnt_modes) * m_deltaO) *...
                    phiaOSET(:,icnt_modes);
            end; % end "for icnt_modes" inner loop

        end;  % End "for icnt_dv" outer loop for sensitivity calculations
        displacmentPlot_OSET % calls a program

        % This program assembles the beam displacement vector for
        % sens_beam(dispX) and base beam(dispA) under ABC

        if  dispX_tot  == 0; % when the vector is empty
            dispX_tot = disp1; % displacement vector used in plotting
            dispA_tot = disp1a;
        else
            dispX_tot = cat(1,dispX_tot,disp1);
            dispA_tot = cat(1, dispA_tot,disp1a);
        end % end "if dispX_tot  == 0"

    end;    % end "if mass_lbls = 0"


    % *******************************************************************
    % ************** EI SENSITIVITY CALCULATION LOOP ********************
    % *******************************************************************
    sens_EIO = 0;
    if EI_lbls ~= 0;

        for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

            %   Resetting BeamX properties to BeamA properties
            element_EI(:,2)  =  element_EI(:,1);

            %   each element, one at a time will have a change in EI
            element_EI(icnt_dv,2) = element_EI(icnt_dv,2) * ...
                (1 + (EI_change/100) );

            Assemble2Beams_rla;   % Run script to assemble beams.

            % Artifical Boundary Conditions
            maO_base = ma(BCoset, BCoset);
            mxO = mx(BCoset, BCoset);

            plotmx = mx_beam(BCoset, BCoset);
            plotkx = kx_beam(BCoset, BCoset);

            kaO_base = ka(BCoset, BCoset);
            kxO = kx(BCoset, BCoset);

            % lam (natural freq^2, rad^2/sec^2), phi (mode shapes)
            [lamaOSET,phiaOSET] = fModes(kaO_base,maO_base);
            [lamxOSET,phixOSET] = fModes(kxO,mxO); %sens info
            [lamxplot,phixplot] = fModes(plotkx,plotmx); %plot info


            if length(pinned)==1 & pinned == 3%Special case of DOF #3 pinned
                phiAPLOT(2:ndof-2, :) = phiaOSET(1:ndof-3, :);
                phiXPLOT(2:ndof-2, :) = phixplot(1:ndof-3, :);

           elseif length(pinned)>1 & pinned(1,1)==3
                phiAPLOT(2:pinned(1,2)-3,:) = phiaOSET(2:pinned(1,2)-3,:);
                phiAPLOT(pinned(1,2)-1:ndof-3,:) = phiaOSET(pinned(1,2)-2:ndof-
4,:);

            elseif length(pinned)<2 %For single ABC
                phiAPLOT(1:pinned-3, :) = phiaOSET(1:pinned-3, :);
                phiAPLOT(pinned-1:ndof-2,:) = phiaOSET(pinned-2:ndof-3,:);
                phiXPLOT(1:pinned-3, :) = phixplot(1:pinned-3, :);
                phiXPLOT(pinned-1:ndof-2,:) = phixplot(pinned-2:ndof-3,:);

            else %For two ABC's
                phiAPLOT(1:pinned(1,1)-3, :) = phiaOSET(1:pinned(1,1)-3, :);
                phiAPLOT(pinned(1,1)-1:pinned(1,2)-3,:) = phiaOSET(pinned(1,1)-
1:pinned(1,2)-3,:);
                phiAPLOT(pinned(1,2)-1:ndof-3,:) = phiaOSET(pinned(1,2)-2:ndof-
4,:);
                phiXPLOT(1:pinned(1,1)-3, :) = phixplot(1:pinned(1,1)-3, :);
                phiXPLOT(pinned(1,1)-1:pinned(1,2)-3,:) = phixplot(pinned(1,1)-
1:pinned(1,2)-3,:);
                phiXPLOT(pinned(1,2)-1:ndof-3,:) = phixplot(pinned(1,2)-2:ndof-
4,:);
           end
            num_rbmOSET = length(find(lamaOSET < 1));
            start_mode = num_rbmOSET + 1;   % Skip the rigid body modes.

            faO = sqrt(lamaOSET)/(2*pi); %natural freq of the ABC, Hz
            %    Form EI derivative matrices:
            k_deltaO = (kxO - kaO_base)/(EI_change/100);   % in %/100

            %    Mode freq sens loop:
            end_mode = start_mode + (num_modesO - 1);
            row_num = 0;

            for icnt_modes = start_mode:end_mode;
                row_num = row_num +1;
                sens_EIO(row_num,icnt_dv) = phiaOSET(:,icnt_modes)' *...
                    k_deltaO * phiaOSET(:,icnt_modes);
            end; %end "for icnt_modes"


        end; % End "for icnt_dv" outer loop for sensitivity calculations


        displacmentPlot_OSET % calls a program

        % This program assembles the beam displacement vector for
        % sens_beam(dispX) and base beam(dispA) under ABC

        if  dispX_tot  == 0;
            dispX_tot = disp1;
            dispA_tot = disp1a;
        else
            dispX_tot = cat(1,dispX_tot,disp1);
            dispA_tot = cat(1, dispA_tot,disp1a);
        end % end "if dispX_tot  == []"

    end;    % end "if EI_lbls = []"


    % Copy element EI and mass properties back into arrays:
    element_EI   = element_EI_orig;
    element_mass = element_mass_orig;

    clear element_EI_orig element_mass_orig end_mode start_mode
    clear row_num icnt_dv icnt_modes lbls num_EI_dv num_mass_dv

    % assemble of total sensitivity matrix for ABC System
    if sens_massO == 0 & sens_EIO ~=0;

        T_sensO = sens_EIO; % no changes in mass

    elseif sens_massO ~= 0 & sens_EIO == 0;
        T_sensO = sens_massO; % no changes in EI

    else
        T_sensO = cat(2, sens_massO,sens_EIO);
        % changes in both mass and EI

    end %end "if sens_massO == 0 & sens_EIO ~=0"

    % Builds complete sens matrix of all ABC systems
    if  T_sens_oset == 0;% when matrix is originally empty

        T_sens_oset = T_sensO;

    else % after the matrix has some values
        T_sens_oset = cat(1,T_sens_oset, T_sensO);
    end %end "if  T_sens_oset == []"

    lamaOSET = lamaOSET(find(lamaOSET > 1)); %skips Rigid body modes
    vect_lamO = lamaOSET(1:num_modesO); % nat freq of base beam under ABC

    %builds a vector of natural freq of ABC systems
    if vect_lam_oset == 0;% when matrix is originally empty


        vect_lam_oset = vect_lamO;
    else% after the matrix has some values
        vect_lam_oset = cat(1, vect_lam_oset, vect_lamO);
    end %end "if vect_lam_oset == 0"
    disp(' ')
    oset_choice = input('  Another cycle of ASET and OSET (y/n)? ','s');
    % loop runs until "n" is inputed creating sens matrix & lam vector
    % for all ABC systems
    disp(' ')

end; % end "while oset_choice ~='n'"

% ********************  BeamSensitivityOSET_rla.m  ***********************
```

G. BOUNDARYCONDITIONS_RLA

```
% *******************  BoundaryConditions_rla.m  **********************

% Written by Prof Gordis

% This script prompts the user boundary condition information
% The script creates a vector of DOF (with respect to the unrestrained
% structure) and then extracts the rows and columns of the complementary
% DOF.

%  Script defines vector "free_dof_set" containing
%  list of unrestrained dof.

% The boundary conditions are applied in this script.

% Inputs needed:
% -------------
% ndof
% ka, ma, kx, mx

% Outputs:
% -------------
% free_dof_set
% updated ka, ma, kx, mx
% icnt_dof
% add_dof
% bc_node
% bc_coord
% bc_DOF
% bc_boolean
% all_dofs
% restraint_switch
% ****************************************************
% Start code:


if exist('free_dof_set')==0;    %  Build free_dof_set vector

    disp(' Select a boundary condition set:')
    disp('    (1) Clamped-free')
    disp('    (2) Clamped-Clamped')
    disp('    (3) Pinned-Pinned')
    disp('    (4) User-Defined')
    disp('    (5) Free-Free')

    BC_Choice = input(' >> Enter choice: ');

    if BC_Choice == 1;      % Clamped-free _____________________________
        free_dof_set = [3:ndof];
        restraint_switch = 'y';

    elseif BC_Choice == 2;  % Clamped-Clamped __________________________

        free_dof_set = [3:ndof-2];
        restraint_switch = 'y';

    elseif BC_Choice == 3;  % Pinned-Pinned ____________________________

        free_dof_set = [2:ndof-2  ndof];
        restraint_switch = 'y';

    elseif BC_Choice == 4;  % User-Defined _____________________________

        icnt_dof = 0;
        add_dof = 'y';
        while add_dof == 'y';

            bc_node = input(' Node number for restraint ? "0" to end: ');

            if bc_node == 0;
                break
            end;
            bc_coord = input(' Translation or Rotation ? (t/r) ','s');

            icnt_dof = icnt_dof + 1;
            if bc_coord == 't';
                bc_DOF(icnt_dof) = 2 * bc_node - 1;
            elseif bc_coord == 'r';
                bc_DOF(icnt_dof) = 2 * bc_node;
            end;   % End if-else block

        end;  % End while add_dof

        bc_boolean = ones(ndof,1);                   % [1 1 1 ... icnt_dof]
        bc_boolean(bc_DOF) = zeros(length(bc_DOF),1);% Put zeros in restrained dof
        all_dofs = [1:ndof];                         % List of all dof
        free_dof_set = all_dofs(logical(bc_boolean));% Extract free dof
        restraint_switch = 'y';

    elseif BC_Choice == 5; % Free-free beam _______________________________

        free_dof_set = [1:ndof];
        restraint_switch = 'n';

    end;                    % End if-elseif choice block __________________

end;      % End exist block

ka = ka(free_dof_set,free_dof_set);
ma = ma(free_dof_set,free_dof_set);
kx = kx(free_dof_set,free_dof_set);
mx = mx(free_dof_set,free_dof_set);

% ****************END BoundaryConditions_rla.m  ***********************
```

H. BUILD2BEAMS_RLA

```
% ********************  Build2Beams_rla.m  **********************

clear
clc
% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 9/22/94  Basic two beam assembly
%       2.0:          Added multi-element changes
%       2.1  3/28/95  Added read/write to file, rebuild capability
%       2.2  3/29/95  Added lumped mass additions
%            3/10/04  Added Sensitivity matrices, error prediction, plots

% ***********************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~~
%
%  This program assembles the mass and stiffness matrices for 2 free-free
%  beams, referred to as "BeamA" (analysis) and "BeamX" (experimental). The
%  program can be run in several modes:
%
%  "Build" mode:
%   ~~~~~  ~~~~~
%  The user provides baseline data for BeamA, assumed to be a
%  homogeneous, uniform beam. Data provided:
%
%     (1) Beam length
%     (2) Number of elements
%     (3) Nominal EI
%     (4) Nominal cross-sectional area
%     (5) Nominal weight density
%
%  The program then prompts the user for instructions on how to modify
%  "BeamA" data to arrive at "BeamX" data. The user can modify element
%  masses, and/or element EI values. The modification can be applied to
%  either a single element, or range of elements, e.g.
%
%      Modify single/range element mass values (y/n)?  y
%
%  If "y" is entered, the user enters the number of the element for mass
%  adjustment:
%
%       Enter element label(s) for mass modification:  1
%       Use MATLAB vector format> 1 3 5:7 9
%
%               Enter percentage mass change (+/- %)
%
% The user is prompted to modify another element or range of elements:
%
%       Modify another element mass value (y/n)?  y
%
%  This process continues until the user enters an "n" for no change.
%  This entire process can then be repeated for EI adjustment.
%
%  The program saves the beam definition data in a binary (.mat) file
%  "beamdata" at the end of execution.
%
%  The program can also be run in "Read" mode by entering an "r" at
%  the initial prompt.
%
%
% Script Execution Path:
% ~~~~~~ ~~~~~~~~~ ~~~~~
%
%
%
%     Build2Beams_crs.m         -- User executes this program.
%     BeamA_Prompt_crs.m        -- Prompts User for BeamA nominal beam data
%     BeamX_Prompt_crs.m        -- Prompts User for BeamX modification beam data
%     Assemble2Beams_crs.m      -- Called by Build2Beams, builds [ka] [ma] [kx]
%                             [mx], plots freqs.
%     AddLumpmass_crs.m            -- Prompts User for BeamX lumped mass addition
%     BoundaryConditions_crs.m  -- Prompts user for B.C.'s and applies them.
%     PlotBeamModes_crs.m       -- Calculate beam modes and plot frequencies
%
```

```
%     BeamSensitivity_crs.m     -- Calculate sensitivity matrix T-sens
%     BeamSensitivityOSET_crs.m -- Calculate sensitivity matrix using ABC
%     recorded_H_crs.m          -- Calculates the nat. freq of BeamX with ABC
applied
%     AssembleSens_crs.m        -- Assembles the sens matrices and calculates
errors.
%     ABCrunTHRU.m              -- Calculates the DV and cond number of matrix used
%     Saves data to "beamdata.mat"

%
%  Start code:
%  ~~~~~ ~~~~~
% *************************************************************************


disp('   Building 2 beams from scratch...')

BeamA_Prompt_rla;       %      Prompt for BeamA Data: run prompt script
BeamX_Prompt_rla;       %      Prompt for BeamX Modification Data:
Assemble2Beams_rla;     %      Run script to assemble mass and stiffness matrices
AddLumpmass_rla;        %      BeamX lumped mass vector construction and
%                           application

kx_beam = kx;  % saves the Beam X matrices without BC to be used later
mx_beam = mx;

BoundaryConditions_rla; %      Prompt for, and apply boundary conditions

kx_beamBC = kx;  % saves the Beam X matrices with BC to be used later
mx_beamBC = mx;
ka_beamBC = ka;  % saves the Beam A matrices with BC to be used later
ma_beamBC = ma;

PlotBeamModes_rla       %  Calculate beam modes and plot frequencies

BeamSensitivity_rla;    %  Calculate sensitivity matrix T-sens
BeamSensitivityOSET_rla;%  Calculate sensitivity matrix using ABC


recorded_H_rla;    % Calulates the nat. freq of BeamX with ABC applied
AssembleSens_rla;  % Assembles the sens matrices and calculates errors.
ABCrunTHRU_rla;    % Calculates the DV and cond number of matrix used
FOM_rla;            %Calculates the Figure of Merit for each prediction


%       Save Defining Parameters for Beams and plots
disp(' ...saving beam data to file')
save beamdata.mat


disp(' Build2Beams end.')
% ______________________

% *******************  END Build2Beams_rla.m  **********************
```

I. DAMAGE_PREDICTOR

```
% % Mode error predictor
%
%
%
format long
%///////////ORGINAL METHOD////////////////////
%calculates dv for base system summed
for q=1:20
     dv_calculated_base(:,q) = T_sens_tot(1:q,:)\vect_lam_tot(1:q);
end

%calculates dv for ABC1 system summed
for t=40:59
    dv_calculated_base1(:,(t-39)) = T_sens_tot(40:t,:)\vect_lam_tot(40:t);
end

%calculates dv for ABC2 system summed
for t=60:79
    dv_calculated_base2(:,(t-59)) = T_sens_tot(60:t,:)\vect_lam_tot(60:t);
end

%calculates dv for ABC3 system summed
for t=80:99
    dv_calculated_base3(:,(t-79)) = T_sens_tot(80:t,:)\vect_lam_tot(80:t);
end

for t=100:119
    dv_calculated_base3(:,(t-99)) = T_sens_tot(100:t,:)\vect_lam_tot(100:t);
end

%For ABC System 1 and 2   %%%%%%%%%%%%%%%%%%%%%%%%%%yo lo comentee


% %%%%%%%%%%%%%%  2 ABC  system   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% T_combo1 = zeros(9,20);
% T_combo1(1:3,:)= T_sens_tot(1:3,:);
% T_combo1(4:6,:)= T_sens_tot(40:42,:);
% T_combo1(7:9,:)= T_sens_tot(78:80,:);
%
% vect_combo1 = zeros(9,1);
% vect_combo1(1:3,1) = vect_lam_tot(1:3,1);
% vect_combo1(4:6,1) = vect_lam_tot(40:42,1);
% vect_combo1(7:9,1) = vect_lam_tot(78:80,1);
%
% for e = 1:9
% pcombo1(e,:) = T_combo1(1:e,:)\vect_combo1(1:e,1)
% end

% bbbb = (pcombo1)'

% %%%%%%%%%%%%  3 ABC  system  %%%%%%%%%%%%%%%%%%%%%%%%
% T_combo1 = zeros(12,20);
% T_combo1(1:3,:)= T_sens_tot(1:3,:);
% T_combo1(4:6,:)= T_sens_tot(40:42,:);
% T_combo1(7:9,:)= T_sens_tot(78:80,:);
% T_combo1(10:12,:)= T_sens_tot(117:119,:);
%
% vect_combo1 = zeros(12,1);
% vect_combo1(1:3,1) = vect_lam_tot(1:3,1);
% vect_combo1(4:6,1) = vect_lam_tot(40:42,1);
% vect_combo1(7:9,1) = vect_lam_tot(78:80,1);
% vect_combo1(10:12,1) = vect_lam_tot(117:119,1);
%
% for e = 1:12
% pcombo1(e,:) = T_combo1(1:e,:)\vect_combo1(1:e,1)
% end
%
% bbbb = (pcombo1)'

%%%%%%%%%%%  5 ABC  system  %%%%%%%%%%%%%%%%%%%%%%%%%%%
T_combo1 = zeros(18,20);
T_combo1(1:3,:)= T_sens_tot(1:3,:);
T_combo1(4:6,:)= T_sens_tot(40:42,:);
T_combo1(7:9,:)= T_sens_tot(78:80,:);
T_combo1(10:12,:)= T_sens_tot(117:119,:);
T_combo1(13:15,:)= T_sens_tot(156:158,:);
```

```
T_combo1(16:18,:)= T_sens_tot(195:197,:);

vect_combo1 = zeros(18,1);
vect_combo1(1:3,1) = vect_lam_tot(1:3,1);
vect_combo1(4:6,1) = vect_lam_tot(40:42,1);
vect_combo1(7:9,1) = vect_lam_tot(78:80,1);
vect_combo1(10:12,1) = vect_lam_tot(117:119,1);
vect_combo1(13:15,1) = vect_lam_tot(156:158,1);
vect_combo1(16:18,1) = vect_lam_tot(195:197,1);

for e = 1:18
pcombo1(e,:) = T_combo1(1:e,:)\vect_combo1(1:e,1)
end

bbbb = (pcombo1)'



% %For base and ABC System    %%%%%%%%%%%%%%%%%%%%%%%%%%yo lo comentee
%
%
% %ABC 1 system
% T_combo1 = zeros(6,20);
% T_combo1(1:3,:)= T_sens_tot(1:3,:);
% T_combo1(4:6,:)= T_sens_tot(40:42,:);
%
% vect_combo1 = zeros(6,1);
% vect_combo1(1:3,1) = vect_lam_tot(1:3,1);
% vect_combo1(4:6,1) = vect_lam_tot(40:42,1)
%
% for e = 1:6
% pcombo1(e,:) = T_combo1(1:e,:)\vect_combo1(1:e,1)
% end
%
% bbbb = (pcombo1)'
%
% %ABC 2 system
% T_combo2 = zeros(6,20);
% T_combo2(1:3,:)= T_sens_tot(1:3,:);
% T_combo2(4:6,:)= T_sens_tot(78:80,:);
%
% vect_combo2 = zeros(6,1);
% vect_combo2(1:3,1) = vect_lam_tot(1:3,1);
% vect_combo2(4:6,1) = vect_lam_tot(78:80,1)
%
% for e = 1:6
% pcombo2(e,:) = T_combo2(1:e,:)\vect_combo2(1:e,1)
% end
% ccccc = cond(T_combo)
% cccc = (pcombo2)'



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%yo lo comentee
```

J. DISPLACEMENTPLOT_OSET_RLA

```
% ********************  displacementPlot_OSET_rla.m  **********************

% Written by Constance Fernandez Spring 2004
% Updated by John Mentzer, Spring 2007

% This program plots the mode shapes (phi, lam) phi vs nodal position
% of beam when ABC are applied.

% Inputs
% -------
% plotkx
% kaO_base
% num_modesO
% phiXPLOT
% phiAPLOT
% pinned
% num_elements

% Outputs
% --------
%disp1, displa
%jj, g
% ypos

%------
% disp1 = zeros(ceil(.5*size(plotkx,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
disp1 = zeros(ceil((20/38)*size(plotkx,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
% disp1a = zeros(ceil(.5*size(kaO_base,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
disp1a = zeros(ceil((20/38)*size(kaO_base,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
for jj = 1:ceil((20/38)*size(plotkx,1));
    disp1(jj+1,:) = phiXPLOT(2*jj-1,1:num_modesO);
    % every other phi to give displacement at sequential nodes
    disp1a(jj+1,:) = phiAPLOT(2*jj-1,1:num_modesO);
end
% This loop normalizes the modes shapes to the tip modal displacement.
if pinned == 41 % tip pinned is a special case,  no new calculations are needed
   disp1(:,:) = disp1(:,:);
   disp1a(:,:) = disp1a(:,:);
elseif length(pinned)>1 & pinned(1,2)==41 %tip pinned with 2 ABC's
   disp1(:,:) = disp1(:,:);
   disp1a(:,:) = disp1a(:,:);
else
for g = 1:num_modesO
   disp1(:,g) = disp1(:,g)/disp1(num_elements+1,g);
   disp1a(:,g) = disp1a(:,g)/disp1a(num_elements+1,g);
end
end
% end
%ypos = [1:1:num_elements+1]; % Location of nodes used in plotting
ypos = [1:1:num_elements+1]; % Location of nodes used in plotting
% if mass_lbls ~= [];
%
%     for kk = 1:size(mass_lbls,1);
%         ff =0;
%         for JJ = 1:length(find(mass_lbls(kk,:)>0));
%             ff = ff+1;
%             posm(kk, 2*JJ-1) = mass_lbls(kk, ff);
%             posm(kk, 2*JJ) = mass_lbls(kk,ff)+1;
%         end
%     end
%
%     if kk == 1
%         posM = posm;
%
%     else
%
%         for uu = 1:kk-1;
%             posM = cat(2, posm(uu,:), posm(uu+1,:));
%         end
%
%     end
%     posM = sort(posM(find(posM>0)));
%     m = .5*ones(size(posM))+icnt_oset;
% end
%
% if EI_lbls ~= [];
%     for kk = 1:size(EI_lbls,1);
%         ff =0;
%         for JJ = 1:length(find(EI_lbls(kk,:)>0));
%             ff = ff+1;
%             pose(kk, 2*JJ-1) = EI_lbls(kk, ff);
%             pose(kk, 2*JJ) = EI_lbls(kk,ff)+1;
%         end
%     end
%
%     if kk == 1
%         posE= pose;
%
%     else
%
%         for uu = 1:kk-1;
%             posE = cat(2, pose(uu,:), pose(uu+1,:));
%         end
%     end
%   posE = sort(posE(find(posE>0)));
%   e = -.5*ones(size(posE))+icnt_oset;
% end


% ********************  END displacementPlot_OSET_rla.m  ***********************
```

K.   FBEAMKM_RLA

```
% % ********************  fbeamkm_rla.m  ***********************
%
% % function [kbeam,mbeam]=fbeamkm(l,ei,m)
% % Provided by Prof Gordis
%
% function [kbeam,mbeam]=fbeamkm(l,ei,m)
% %
% %
% % This function returns the stiffness and mass matrices for
% % a simple 2-node beam element.
% %
% % Note: m = rho * area * length = total element mass
% %
% % Reference: R.D. Cook, Concepts and Applications of F.E. Analysis
%
% % Outputs
% % ------
% % kbeam, mbeam, i, j
%
%
% kbeam=zeros(4,4);
% mbeam=zeros(4,4);
% %
% kbeam(1,1)=12.0;
% kbeam(1,2)=6.0*l;
% kbeam(1,3)=-12.0;
% kbeam(1,4)=6.0*l;
% kbeam(2,2)=4.0*l^2;
% kbeam(2,3)=-6.0*l;
% kbeam(2,4)=2.0*l^2;
% kbeam(3,3)=12.0;
% kbeam(3,4)=-6.0*l;
% kbeam(4,4)=4.0*l^2;
% %
% mbeam(1,1)=156.0;
% mbeam(1,2)=22.0*l;
% mbeam(1,3)=54.0;
% mbeam(1,4)=-13.0*l;
% mbeam(2,2)=4.0*l^2;
% mbeam(2,3)=13.0*l;
% mbeam(2,4)=-3.0*l^2;
% mbeam(3,3)=156.0;
% mbeam(3,4)=-22.0*l;
% mbeam(4,4)=4.0*l^2;
% %
% for i=1:4;
%     for j=i:4;
%         kbeam(j,i)=kbeam(i,j);
%         mbeam(j,i)=mbeam(i,j);
%     end
% end
% %
% kbeam=(ei/l^3)*kbeam;
% mbeam=(m/420.0)*mbeam;
% %
% % end function beamkm
%
% % ********************  END fbeamkm_rla.m  ***********************
```

L. FMODES

```
% % ********************  fMode s.m  ***********************
%
% function [lam,phi]=fmodes(k,m,num_to_print);
% % Provided by Prof Gordis
% % This program prints to the screen natural modes of system (phi).
% %
% %     Usage: [lam,phi]=fmodes(k,m,num_to_print)
% %
% %   This function can be used with 1 to 3 arguments, as follows:
% %
% %     [lam,phi]=fmodes(a)   : Produces modes of [a] with no print of freqs in
Hz.
% %     [lam,phi]=fmodes(a,i) : Produces modes of [a] with print of "i" freqs in
Hz.
% %     [lam,phi]=fmodes(k,m) : Produces modes of [m\k] with no print of freqs in
Hz.
% %
% %
% %
% %     This function returns a vector containing eigenvalues (rad/sec)^2
% %     and a matrix containing the mass normalized mode shapes.
% %     The mode information is sorted by frequency in ascending order.
% %     If num_to_print > 0; tabular listing of num_to_print freqs in Hz is
printed.
% %     If num_to_print <= 0, no print.
%
% % Inputs
% % ------
% % v, index, m, k
%
% % Programs
% % --------
% % fNormalize
%
% % Outputs
% % --------
% % phi
% % num_to_print
% % error
% % e
%
% %     ---------------------------------------------------------------------------
-----
%
% if nargin == 1;
 %      [A] w/ no print request for freqs in Hz.
%
        %   v(1,:) = 1 normalization
%       [v,d]=eig(k);
%       [temp,indices] = sort(abs(diag(d)));
%       lam = diag(d);
%       lam = lam(indices);
%       [phi]=fNormalize(v(:,indices), 'one');
%       num_to_print = 0;
%
% elseif nargin == 2 & size(m,1) == 1;                      %      [A] w/ print
request for freqs in Hz.
%
        %   v(1,:) = 1 normalization
%       [v,d]=eig(k);
%       [temp,indices] = sort(abs(diag(d)));
%       lam = diag(d);
%       lam = lam(indices);
%       [phi]=fNormalize(v(:,indices), 'one');
%       num_to_print = m;
%
% elseif nargin == 2 & size(m,1) > 1;                              %      [k],[m]
w/ no print request for freqs in Hz.
%
        %   mass normalization
%       [v,d]=eig(m\k);
%       [lam,index]=sort(abs(diag(d)));
%       [phi]=fNormalize(v(:,index),'mass',m);
%       num_to_print = 0;
%
```

```
% elseif nargin == 3 & size(k,1) > 1 & size(m,1) > 1;%      [k],[m] w/ print
request for freqs in Hz.
%
        %   mass normalization
%       [v,d]=eig(m\k);
%       [lam,index]=sort(abs(diag(d)));
%       [phi]=fNormalize(v(:,index),'mass',m);
%
% else
%
%       num_to_print = -1;
%       error('Error from fModes.m: Check input arguments.')
%
% end
%
% if num_to_print > length(k);
%       num_to_print = length(k);
% end
%
% if nargin < 3 & rem(length(k),2)==0 & k(1:length(k)/2,1:length(k)/2) ==
zeros(length(k)/2,length(k)/2); % Have [A] matrix
%       e = 1;         % Eigenvalues are wn
% else
%       e = 0.5;    % Eigenvalues are wn^2
% end
%
%
%
% if num_to_print > 0;
%
%       disp('  '),disp('  ')
%       disp('~~~~~~~~~~~~~')
%       disp('Freqs in Hz.:')
%       disp((lam(1:num_to_print).^e)/2/pi)
%       disp('~~~~~~~~~~~~~')
%
% end
%
% % ********************  END fModes.m  ***********************
```

M. FNORMALIZE

```
% % *******************  fNormalize.m  **********************
%
% function [phi] = fNormalize(phi,method,m);
% %
% % Usage: [phi] = fNormalize(phi,method,m);
% %
% %     phi: matrix whose columns are to be (independently) normalized.
% %     method: String variable. The following choices are available:
% %
% %                   'mass'                Mass normalization
% %                   'inf'                 Infinity normalization
% %                   'one'                 First element = 1
% %                   'length'              Length = 1
% %
% %     m: matrix used for normalization, i.e., phi'*m*phi = eye
% %
% % _________________________________________
% %
%       switch method
%
%              case 'mass'                  % Mass normalization
%
% %                   disp('mass normalization')
%                     phi = phi * diag(sqrt(diag((phi' * m * phi).^(-1))));
%
%              case 'inf'                   % Infinity normalization
% %                   disp('inf normalization')
%                     for icnt_cols = 1:size(phi,2);
%                            phi(:,icnt_cols) =
phi(:,icnt_cols)/norm(phi(:,icnt_cols),inf);
%                     end
%
%              case 'one'                   % First element = 1
% %                   disp('one normalization')
%                     phi = phi * diag((phi(1,:).^(-1))');
%
%              case 'length'         % Length = 1
% %                   disp('length normalization')
%                     for icnt_cols = 1:size(phi,2);
%                            phi(:,icnt_cols) =
phi(:,icnt_cols)./norm(phi(:,icnt_cols),'fro');
%                     end
%
%       end
%
% % *******************  END fNormalize.m  **********************
```

N. FOSET_FROM_ASET

```
% % *******************  fOset_from_Aset.m  **********************
%
% function [oset] = fOset_from_Aset(ndof,aset);
% %
% %  Usage: [oset] = fOset_from_Aset(ndof,aset);
% %
% % This function determines the complementary subset "oset"
% % from a set [1:1:ndof] and the subset aset = [x x x ...].
% %
% %  ndof: Total number of DOF. Set is labeled "nset".
% %  aset: Retained DOF (proper subset of [1:1:ndof])
% %  oset: aset U oset = n
% %
% % Provided by Prof Gordis
% % ________________________________________________________
%
% nset = [1:ndof];
%
% for icnt = 1 : length(aset);
%      indices(icnt) = find(nset == aset(icnt));
% end
%
% bool = ones(size(nset));
% bool(indices) = zeros(size(indices));
% oset = nset(find(bool>0));
%
% % *******************  fOset_from_Aset.m  **********************
```

## O.	FSPRINGMASS2

```
% % *******************  fSpringMass2.m  **********************
%
% function [k,m]=fSpringMass2(springs,mass,BC);
% %
% %      Usage: function [k,m]=fSpringMass2(springs,mass,BC)
% %
% %      This function script assembles the stiffness [k] and mass
% %      [m] matrices for an assemblage of springs.
% %
% %
% %      A linear chain of springs and masses is assumed.
% %             The number of springs is defined by the length of the vector
'springs'
% %             and their values by the elements of 'springs.'
% %
% %             The number of masses is defined by the length of the vector 'mass'
% %             and their values by the elements of 'mass'.
% %             NOTE:  The number of masses must equal to the final number of
active
% %                           DOF (i.e., after BC's applied).
% %
% %      Boundary conditions are specified by the vector 'BC.' This vector
% %      contains the DOF numbers which are to be restrained.
% %
% %      For example, to build the following system:
% %
% %                      .01        .02       .015
% %               |--////--[m]--////--[m]--////--[m]
% %                    5        6       3.4
% %
%       springs = [1 1 ];
%       mass    = [.01 .01 ];
%       BC             = [1]
% %
% %
% % _________________________________________________________
% %
% %                          BEGIN SCRIPT
% %                          ~~~~~ ~~~~~~
% %
%
% if length(mass) == (length(springs)+1) - length(BC);
%
%       k              = zeros(length(springs)+1,length(springs)+1);
%       m              = zeros(length(mass));
%
% %      assemble stiffness matrix:
%
%       rows = [0 1];
%       for ispring = 1 : length(springs);
%
%              rows = rows + 1;
%
%              addthis = [springs(ispring) -springs(ispring);-springs(ispring)
springs(ispring)];
%              k(rows,rows) = k(rows,rows) + addthis;
%       end
%
%       if ~isempty(BC);
%              keep = fOset_from_Aset(length(springs)+1,BC);
%              k = k(keep,keep);
%       end
%
% %      assemble mass matrix:
%
%       m = diag(mass);
%
% else
%
%       disp('Error in fSpringmass2. Check # masses, springs, and BC"s.')
%       return
%
% end
% % *******************  END fSpringMass2.m  **********************
%
```

P. KINETICENERGY

```
% This code solves for the Kinetic Energy function for structural systems

clc
format long

%Given data
% E = 10e6;
% I1 = 0.016067021;
% nominal_area
%Element Lenght
% rho
L = total_length/num_elements;

% Making my eigenvector matrix a 42 X number of modes required
a = zeros(42,40);
a(3:42,:) = phi_base(:,:);

% For loops generated to select the corresponding eigenvectors (magnitudes
% and displacements) out of the phi_base matrix and then multiplied them
% with the corresponding hermitian shape function second derivatives.
for i = 1:num_elements
   for k = 1:ndof-2
displeft(i,k) = a(2*i - 1,k);
rotleft(i,k) =  a(2*i,k);
dispright(i,k) = a(2*i + 1,k);
rotright(i,k) =  a(2*i + 2,k);
%After integrating the original Hermitians Shape functions squared, I
%obtain jprime:
jprime(i,k) = 1/7*(1/L^2*rotleft(i,k)+2/L^3*displeft(i,k)-
2/L^3*dispright(i,k)+1/L^2*rotright(i,k))^2*L^7+1/3*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))*(1/L^2*rotleft(i,k)+2/L^3*displeft(i,k)-
2/L^3*dispright(i,k)+1/L^2*rotright(i,k))*L^6+1/5*(2*rotleft(i,k)*(1/L^2*rotleft(i,
k)+2/L^3*displeft(i,k)-2/L^3*dispright(i,k)+1/L^2*rotright(i,k))+(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))^2)*L^5+1/4*(2*displeft(i,k)*(1/L^2*rotleft(i,k)+2/L^3*displeft(i
,k)-2/L^3*dispright(i,k)+1/L^2*rotright(i,k))+2*rotleft(i,k)*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k)))*L^4+1/3*(2*displeft(i,k)*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))+rotleft(i,k)^2)*L^3+displeft(i,k)*rotleft(i,k)*L^2+displeft(i,k)
^2*L;

f = lam_base';
 kin_lam(i,k)= f(:,k);
 kin_lamsquared = (kin_lam).^2;

   end
end

T = (nominal_area*rho*kin_lamsquared/2).*jprime
```

Q. KINETICENERGYABC

```
%%%%Kinetic Energy calculation of the base beam, with one ABC applied, not
sitffnes/mass perturbations.

clc
format long

%Given data
%E = 10e6;
%I1 = 0.016067021;
L = total_length/num_elements;

%Making my eigenvector matrix a 42 X number of modes required
aa = zeros(42,39);
Keep = [3:pinned - 1,pinned + 1:42];%This for 20 elements. The 42 (ndof) is to be
changed if # elements changes
aa(Keep,:) = phiaOSET(:,:);
%For loops generated to select the corresponding eigenvectors (magnitudes
%and displacements) out of the phi_base matrix and then multiplied them
%with the corresponding hermitian shape function second derivatives.
for i = 1:num_elements
   for k = 1:ndof-3
displeft(i,k) = aa(2*i - 1,k);
rotleft(i,k)  = aa(2*i,k);
dispright(i,k)= aa(2*i + 1,k);
rotright(i,k) = aa(2*i + 2,k);

%"iabcprime" corresponds to the integral part of the strain energy equation
%already calculated in a polynomial form, with the corresponding hermitian second
derivative
%times its corresponding eigenvector product indicated.
jabcprime(i,k) = 1/7*(1/L^2*rotleft(i,k)+2/L^3*displeft(i,k)-
2/L^3*dispright(i,k)+1/L^2*rotright(i,k))^2*L^7+1/3*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))*(1/L^2*rotleft(i,k)+2/L^3*displeft(i,k)-
2/L^3*dispright(i,k)+1/L^2*rotright(i,k))*L^6+1/5*(2*rotleft(i,k)*(1/L^2*rotleft(i,
k)+2/L^3*displeft(i,k)-2/L^3*dispright(i,k)+1/L^2*rotright(i,k))+(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))^2)*L^5+1/4*(2*displeft(i,k)*(1/L^2*rotleft(i,k)+2/L^3*displeft(i
,k)-2/L^3*dispright(i,k)+1/L^2*rotright(i,k))+2*rotleft(i,k)*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k)))*L^4+1/3*(2*displeft(i,k)*(-
3/L^2*displeft(i,k)+3/L^2*dispright(i,k)-2/L*rotleft(i,k)-
1/L*rotright(i,k))+rotleft(i,k)^2)*L^3+displeft(i,k)*rotleft(i,k)*L^2+displeft(i,k)
^2*L;

f = lamaOSET';
 kin_lam(i,k)= f(:,k);
 kin_lamsquared = (kin_lam).^2;


 end
end
% Assembling the strain energy equation:
Tabc = (nominal_area*rho*kin_lamsquared/2).*jabcprime
```

R. MULTIABCS_MODE_ERROR

```
%The purpose of this program is to graph the error prediction wrt number of
%modes retained.  This is done for both base system and the identified ABC.

%This is purely a plotting code


norm_T=zeros(78,20);
for t =1:78;

    norm_T(t,1:20)= T_sens_tot(t,:)/norm(T_sens_tot(t,:),inf);
end

x = 1:20;
figure(101)
title('Error Prediction for Base system'); hold on

subplot(5,1,1)
% bar(x,(p(:,1)),.25,'r');hold on
bar(x,(dv_calculated_base1(:,1)),.25,'b');hold on
bar(x,(dv_calculated_base2(:,1)),.25,'g');hold on
%bar(x,(pabc3(:,1)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
%plot((BC(3,3)/2),0,'g*');hold on
xlim([0 21])
ylim([-.1 .1])
grid on
stem(mass_lbls, dv_mass,'g','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'r','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('MODE 1')
%             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

subplot(5,1,2)
% bar(x,(p(:,2)),.25,'r');hold on
bar(x,(dv_calculated_base1(:,2)),.25,'b');hold on
bar(x,(dv_calculated_base2(:,2)),.25,'g');hold on
%bar(x,(pabc3(:,2)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
%plot((BC(3,3)/2),0,'g*');hold on
xlim([0 21])
ylim([-.1 .1])
grid on
stem(mass_lbls, dv_mass,'g','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'r','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('MODES 1:2')

subplot(5,1,3)
% bar(x,(p(:,3)),.25,'r');hold on
bar(x,(dv_calculated_base1(:,3)),.25,'b');hold on
bar(x,(dv_calculated_base2(:,3)),.25,'g');hold on
%bar(x,(pabc3(:,3)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
%plot((BC(3,3)/2),0,'g*');hold on
xlim([0 21])
ylim([-.1 .1])
grid on
stem(mass_lbls, dv_mass,'g','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'r','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('MODES 1:3')

subplot(5,1,4)
% bar(x,(p(:,4)),.25,'r');hold on
bar(x,(dv_calculated_base1(:,4)),.25,'b');hold on
bar(x,(dv_calculated_base2(:,4)),.25,'g');hold on
% x,(pabc3(:,4)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
```

```
%plot((BC(3,3)/2),0,'g*');hold on
xlim([0 21])
ylim([-.1 .1])
grid on
stem(mass_lbls, dv_mass,'g','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'r','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('MODES 1:4')

subplot(5,1,5)
% bar(x,(p(:,5)),.25,'r');hold on
bar(x,(dv_calculated_base1(:,5)),.25,'b');hold on
bar(x,(dv_calculated_base2(:,5)),.25,'g');hold on
%bar(x,(pabc3(:,5)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
%plot((BC(3,3)/2),0,'g*');hold on
xlim([0 21])
ylim([-.1 .1])
grid on
stem(mass_lbls, dv_mass,'g','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'r','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('MODES 1:5')




% % Base and ABC  system/////////////////////yo lo
comente///////////////////////////////
%
x = 1:20
figure(103)
bar(x,(bbbb(:,9)),.25,'b');hold on
% bar(x,(cccc(:,6)),.25,'g');hold on
plot((BC(1,3)/2),0,'r^',(BC(1,3)/2),0,'rh',(BC(1,3)/2),0,'r*' );hold on
plot((BC(2,3)/2),0,'r^',(BC(2,3)/2),0,'rh',(BC(2,3)/2),0,'r*' );hold on
plot((BC(3,3)/2),0,'r^',(BC(3,3)/2),0,'rh',(BC(3,3)/2),0,'r*' );hold on
plot((BC(4,3)/2),0,'r^',(BC(4,3)/2),0,'rh',(BC(4,3)/2),0,'r*' );hold on
plot((BC(5,3)/2),0,'r^',(BC(5,3)/2),0,'rh',(BC(5,3)/2),0,'r*' );hold on
grid on
xlim([0 21])
ylim([0 .1])
stem(mass_lbls, dv_mass,'y','filled'); hold on;
stem(mass_lbls, dv_mass,'k'), hold on;
stem(EI_lbls, dv_EI,'c','filled');hold on;
stem(EI_lbls, dv_EI,'k'); hold on
title('FIVE ABC SYSTEMS MODES 1:3')
%
```

S. NORMRUNTHRU_CRS

```
% ********************  normRUNthru_crs.m  **********************

% This program finds the NORM of the columns of sensitivity matrix and the
% NORM of the rows of the inverse of the sensitivity matrix.  This was used
% to find a correlation of the good prediction to the ABC system used.  It
% also plots the information in helpful graphes.
%
% This program was written for a system of 19 natural freq. Set of 5 modes
% were used in each ABC system, i.e.,, modes 1-5, modes 6-10,or modes 11-15.
% This accounts for the 3 sets of modes per condition as listed below in
% the "for" loop modeN = 1:3. This program compares the use of the first 5
% modes of the base system and one set of five modes of the ABC  to that of
% of just 10 modes of the ABC.  Notice that the sensitivity is a 10x10
% square matrix indicating that only mass or EI changes, not both were
% made.

% This program is not part of Build2Beams_crs.m program.  It is run
% separately.

% Written by Constance R S Fernandez, Spring 2004

% Inputs
% ------
% cond_basePlus
% icnt_oset
% T_sens_tot
% EI_lbls, mass_lbls
% dv_mass, dv_EI, dv_cal_ABCten

% Outputs
% --------
% BASE
% BASET
% abcN, countN, a_cN
% modeN, startmodeN, startmodeNT
% bb, t, tINV, cc, T, TINV, vv, tt
% modelabelNORM
% norm_vectT, norm_vecTABC, norm_vecTinvABC
% normC
% baseN, baseABCN, abc_conN, abc_conTN
% baseABCNten, abc_conNten, abc_conTNten

% ----Start program----

BASE = int2str(cond_basePlus(1)); % cond no. of the base line system
BASET = sprintf('Base[1:5] Cond = %s', BASE); % used for plotting

% ======Initialization=====%

abcN = 0;
countN =0;

% =======Calculations of NORM vectors======%
for count = 1:icnt_oset +1 % number of conditions (base + ABC)
    a_cN = 1;

    for modeN = 1:3 % 3 sets of modes per boundry condition
        startmodeN = abcN + a_cN; % the beginning mode number of each set

        % indicates the use of the first 5 modes of base system + 5 modes
        % of ABC system
        bb = [1:5, startmodeN: startmodeN+4];

        % labeling of modes for plotting
        modelabelNORM = int2str(a_cN:a_cN+4);

        a_cN = a_cN+5;    %advances to the next set of modes

        % base system plus 5 modes of ABC
        t = T_sens_tot(bb,:); % 10x10 matrix
        tINV = inv(T_sens_tot(bb,:));% 10x10 matrix

        % first 10 modes of ABC solo
        startmodeNT = abcN+1; % the beginning mode number of each set
        cc = [startmodeNT: startmodeNT+9];
        T = T_sens_tot(cc,:);% 10x10 matrix
        TINV = inv(T_sens_tot(cc,:));% 10x10 matrix

        % for loop for NORM of columns and rows of inv(sens matrix)
        for vv = 1:10 % 10 rows, 10 columns
            % base + ABC system
            norm_vecT(vv,countN+modeN) = norm(t(:,vv)); % columns
            norm_vecTinv(vv,countN+modeN) = norm(tINV(vv,:)); % rows
            % for 10 modes ABC solo
            norm_vecTABC(vv,countN+modeN) = norm(T(:,vv)); % columns
            norm_vecTinvABC(vv,countN+modeN) = norm(TINV(vv,:)); % rows

        end % vv loop
    end % ModeN loop
    abcN = abcN+19; % advances to the next ABC system
    countN = countN+3; % counts up each set of ABC
end % count loop
normC=4; % initialize for plots

%==========PLOTTING==========%

for tt=1:10 % figures (30-40) plots 6 graphes per figure
    figure(tt+30)
    subplot(3,2,1)
    bar(norm_vecT(:,normC))
    title ('Norm col Tsens, ABC Modes [1:5]');

    subplot(3,2,3)
    bar(norm_vecTinv(:,normC))

    title ('Norm row TsensINV, ABC Modes [1:5]');

    subplot(3,2,2)
    bar(norm_vecTABC(:,normC))
    title ('Norm col Tsens, ABC Modes [1:10]')
    subplot(3,2,4)
    bar(norm_vecTinvABC(:,normC))
    title ('Norm row TsensINV, ABC Modes [1:10]')

    subplot(3,2,5)
    % plotting error prediction
    % using [1:5] modes of ABC system + [1:5] modes of Base system;
    baseABCN = bar(dv_cal_BasePlus(:,normC),.5,'r');hold on

    % plotting error prediction using [1:5] modes of Base system;
    baseN = bar(dv_cal_ABC(:,1),.25,'b');


    abc_conN = int2str(cond_basePlus(normC)); % cond no. for legend
    abc_conTN = sprintf('Base[1:5]+ABC[1:5] Cond = %s', abc_conN);

    grid on
    legend([baseABCN,baseN],BASET,abc_conTN), hold on


    % plotting actual error
    if EI_lbls ~=[] & mass_lbls ~=[]
        stem(mass_lbls, dv_mass,'y','filled'); hold on;
        stem(mass_lbls, dv_mass,'k')

        EIplot = EI_lbls+10;hold on
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    elseif  mass_lbls ~=[] & EI_lbls ==[]
        stem(mass_lbls, dv_mass,'y','filled');hold on;
        stem(mass_lbls, dv_mass,'k')

    else
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    end % if EI_lbls ~=[] & mass_lbls ~=[]

    title (sprintf('Error, Base [1:5] + ABC [1:5], pinned NODE # %d', (tt+1)))

    subplot(3,2,6)
    % plotting error prediction using [1:10] modes of ABC system;
    baseABCNten = bar(dv_cal_ABCten(:,normC));hold on
    abc_conNten = int2str(cond_ABCten(normC));
    abc_conTNten = sprintf('ABC[1:10] Cond = %s', abc_conNten);

    grid on
    legend([baseABCNten],abc_conTNten), hold on

    % plotting actual error
    if EI_lbls ~=[] & mass_lbls ~=[]
        stem(mass_lbls, dv_mass,'y','filled'); hold on;
        stem(mass_lbls, dv_mass,'k')

        EIplot = EI_lbls+10;hold on
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    elseif  mass_lbls ~=[] & EI_lbls ==[]
        stem(mass_lbls, dv_mass,'y','filled');hold on;
        stem(mass_lbls, dv_mass,'k')

    else
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    end % EI_lbls ~=[] & mass_lbls ~=[]


    title (sprintf('Error, ABC Modes [1:10], pinned NODE # %d', (tt+1)))

    normC = normC+3; % advances to the next ABC system
end % tt = 1:10 for plotting graphes


% ********************  END normRUNthru_crs.m  ***********************
```

T. PLOTBEAMMODES_CRS

```
% *******************  PlotBeamModes_crs.m  **********************

% Calculates natural frequencies

% Provided by Prof Gordis

% Inputs needed:
% -----------------
% ka, ma, mx, kx

% Programs needed:
% -----------------
% fModes

% Outputs:
% -----------------
% lama, phia, lamx, phix (without rigid body modes)
% num_rbm
% phia_plot, phix_plot


disp('  ');
disp(' Calculating modes for each beam...plot frequency comparison')


% Get modes of each beam:

[lama,phia]=fModes(ka,ma);
[lamx,phix]=fModes(kx,mx);

%used to plot the mode shapes org BC before ABC
phia_plot = phia;
phix_plot = phix;

% Set any rigid body mode freqs to zero:

  num_rbm = length(find(lama < 1));

  sprintf('Number of Rigid Body Modes Found: %2i', num_rbm)

    disp( ' Removing rigid body mode frequencies from vectors...')
    lama = lama(find(lama > 1));
    lamx = lamx(find(lamx > 1));

% *******************  END PlotBeamModes_crs.m  **********************
```

U. PLOTTINGBARS_CRS

```
% ********************  plottingBARS_crs.m  **********************

% To be used with Build2Beams.m and
%
% This program plots 9 graphes per figure.  The first columns of 3 graphes
% are the mode shapes of the ABC system used in error prediction.  The next
% column of 3 graphes are the error prediction using only 5 modes of ABC
% system. The last column of 3 graphes are the error predictions using the
% first 5 modes of base system plus 5 modes of the ABC system.  The row
% represent modes 1-5, middle row: modes 6-10, last row : modes 11-15. Each
% of the error prediction graphes also have the base only prediction  and
% the actual error plotted for easy reference.
%
% Written by Constance R S Fernandez, Spring 2004

% Inputs
% --------
% cond_basePlus
% FOM_ABC5per, FOM_PLUSper
% icnt_oset
% modeshape
% rel_freqERROR
% ypos
% EI_lbls, mass_lbls
% dv_mass, dv_EI

% Outputs
% --------
% BASE, BASET
% FOMBASE, FOMABC, FOMPLUS
% intervelp
% ER, barp, shape, error, a_cp, modep, ap,
% modelabelp, FOMABClabelp, FOMPLUSlabelp
% abc_con, abc_conT
% ABC, base
% plus_con, plus_conT
% base, plus, EI_plot


BASE = int2str(cond_basePlus(1));
FOMBASE = int2str(FOM_ABC5per(1));
BASET = sprintf('Base Cond = %s, FOM = %s', BASE, FOMBASE);


intervelp = 3;
modeshape = 1;
ER = 1;
for barp = 1:icnt_oset

    figure(barp+10) % figures 11-20
    format bank
    %shape = [modeshape:modeshape+20]; %This number is equal to number of elements.
    shape = [modeshape:modeshape+20]; %This number is equal to number of elements.
    error = round(rel_freqERROR(ER:ER+15)*100)/100;
    a_cp = 1;
    for modep = 1:3 %3 sets of modes per boundry condition

        ap = [a_cp: a_cp+4]; %modes
%         REL_error1 = int2str(error(a_cp));
%         Errorlabelp = sprintf('Rel error = %s', REL_error1);
%         REL_error2 = int2str(error(a_cp+1));
%          Errorlabelp = sprintf('Rel error = %s', REL_error2);
%         REL_error3 = int2str(error(a_cp+2));
%          Errorlabelp = sprintf('Rel error = %s', REL_error3);
%         REL_error4 = int2str(error(a_cp+3));
%          Errorlabelp = sprintf('Rel error = %s', REL_error4);
%         REL_error5 = int2str(error(a_cp+4));
%          Errorlabelp = sprintf('Rel error = %s', REL_error5);

        modelabelp = int2str(a_cp:a_cp+4);
        FOMABC = int2str(FOM_ABC5per(intervelp+modep));
        FOMABClabelp = sprintf('System FOM = %s', FOMABC);

        FOMPLUS = int2str(FOM_PLUSper(intervelp+modep));
        FOMPLUSlabelp = sprintf('System FOM = %s', FOMPLUS);
        % ============================================================%
        % =====mode shape or beam X and beam A with ABC=================%
        % ============================================================%
        figure(barp+50)
        subplot(3,1,modep)
        plot(ypos, dispA_tot(shape,a_cp),'k-o', ypos, ...
            dispA_tot(shape,a_cp+1),'g-s', ypos, ...
            dispA_tot(shape,a_cp+2),'b-d', ypos, ...
            dispA_tot(shape,a_cp+3),'r-x', ypos, ...
            dispA_tot(shape,a_cp+4),'m-*', ypos, ...
            dispX_tot(shape,a_cp),  'r--o', ypos, ...
            dispX_tot(shape,a_cp+1),'b--s', ypos, ...
            dispX_tot(shape,a_cp+2),'m--d', ypos, ...
            dispX_tot(shape,a_cp+3),'c--x', ypos, ...
            dispX_tot(shape,a_cp+4),'k--*'), grid on...


        legend(sprintf('Rel Freq Error = %d', error(a_cp)),...
            sprintf('Rel Freq Error = %d', error(a_cp+1)),...
            sprintf('Rel Freq Error = %d', error(a_cp+2)),...
            sprintf('Rel Freq Error = %d', error(a_cp+3)),...
            sprintf('Rel Freq Error = %d', error(a_cp+4)));

%         legend(sprintf('Bm X, Md %d', a_cp'), sprintf('Bm X, Md %d', a_cp+1),...
%             sprintf('Bm X, Md %d', a_cp+2), sprintf('Bm X, Md %d',a_cp+3),...
%             sprintf('Bm X, Md %d', a_cp+4), sprintf('Base, Md %d', a_cp), ...
%             sprintf('Base, Md %d', a_cp+1), sprintf('Base, Md %d', a_cp+2), ...
%             sprintf('Base, Md %d', a_cp+3), sprintf('Base, Md %d', a_cp+4))


        title(sprintf('Modes [ %s]',modelabelp))
        axis tight
        % ============================================================%
        % ===== bar graphes of error solution using only ABC=============%
        % ============================================================%
        figure(barp+10) % figures 11-20
        subplot(3,2,2*modep-1)

        abc_con = int2str(cond_ABC(intervelp+modep));
        abc_conT = sprintf('ABC Cond = %s, FOM = %s', abc_con, FOMABC);

        ABC = bar(dv_cal_ABC(:,intervelp+modep),.5,'r'); hold on
        %ABC = bar(dv_cal_ABC(:,1:5),.5,'r'); hold on  %trying to isolate the first
five modes

        base = bar(dv_cal_ABC(:,1),.25,'b');hold off%on % base first 5 modes
        %FOMabc = bar(1,0); hold off
        grid on
        %legend([ABC,base,FOMabc],plus_conT,BASET,FOMABClabelp), hold on
       % grid on
        legend([ABC,base],abc_conT,BASET), hold on

        title(sprintf('ABC only, [ %s]', modelabelp));




        if EI_lbls ~=0 & mass_lbls ~=0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on;
            stem(mass_lbls, dv_mass,'k'), hold on;

            EIplot = EI_lbls+10; hold on % last half of plot
            stem(EIplot, dv_EI,'c','filled');hold on;
            stem(EIplot, dv_EI,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*',...
             %   barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*',...
               ((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-1)/2),0,'g*'
)


        elseif  mass_lbls ~=0 %&EI_lbls =0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on
            stem(mass_lbls, dv_mass,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
            plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        else
            % plots actual error
            stem(EI_lbls, dv_EI,'c','filled');hold on;
            stem(EI_lbls, dv_EI,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
            plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        end % EI_lbls ...

        % ============================================================%
        % =========bar graphes of error solution using ABC + base========%
        % ============================================================%
        subplot(3,2,2*modep)


        plus_con = int2str(cond_basePlus(intervelp+modep));% for legend
        plus_conT = sprintf('Base+ABC Cond = %s FOM = %s', plus_con, FOMPLUS);% for
legend

        plus = bar(dv_cal_BasePlus(:,intervelp+modep),.5,'r'); hold on
        base = bar(dv_cal_ABC(:,1),.25,'b'); hold on % base first 5 modes
        %FOMplus = bar(1,0); hold off
        grid on
        legend([plus,base],plus_conT,BASET), hold on
%        legend([plus,base,FOMplus],plus_conT,BASET,FOMPLUSlabelp), hold on

        if EI_lbls ~=0 & mass_lbls ~=0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on;
            stem(mass_lbls, dv_mass,'k'); hold on
            EIplot = EI_lbls+10; hold on % last half of plot
            stem(EIplot, dv_EI,'c','filled');hold on;
            stem(EIplot, dv_EI,'k');hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*',...
             %   barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )
           plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*',...
               ((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-1)/2),0,'g*'
)

         elseif  mass_lbls ~=0 %&EI_lbls =0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled');hold on;
            stem(mass_lbls, dv_mass,'k');hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        else
            % plots actual error
            stem(EI_lbls, dv_EI,'c','filled');hold on;
            stem(EI_lbls, dv_EI,'k'), hold on
            % plots the green triangle which indicates pinned node
           % plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')%changed barp+1
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        end % if EI_lbls ...

        title(sprintf('Base [1:5] + ABC [ %s]', modelabelp));
        %       title(sprintf('Base + ABC, pinned at NODE# %d',barp +1))
        a_cp= a_cp +5;
    end  % modep loop

    modeshape = modeshape + 11; % advances
    intervelp = intervelp +3; % advances to the next ABC system
    ER = ER+19;
end % barp loop
```

```
%  figure(barp+11)
%plot(1:20,ABC1, bar)


%  %ABC Mode 1 Error
%         subplot(5,1,1)
%         ABC1 =dv_cal_ABC1;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC1,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 1')
% %ABC Mode 2 Error
%         subplot(5,1,2)
%
%         ABC2 = dv_cal_ABC2;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC2,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*') ;
%         title('ABC Mode 2')
%
% %ABC Mode 3 Error
%         subplot(5,1,3)
%
%         ABC3 = dv_cal_ABC3;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC3,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*') ;
%         title('ABC Mode 3')
%
%  %ABC Mode 4 Error
%         subplot(5,1,4)
%         ABC4 =dv_cal_ABC4;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC4,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 4')
%
% %ABC Mode 5 Error
%         subplot(5,1,5)
%         ABC5 =dv_cal_ABC5;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC5,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 5')





% ********************  END plottingBARS_crs.m  ***********************
```

V. STRAINENERGY

```
%%%Strain Energy calculation of the base beam, that is neither ABC's nor
%%%sitffnes/mass perturbations applied.

clc
format long

%Given data
% E = 10e6;
% I1 = 0.016067021;
%Element Lenght
L = total_length/num_elements;

%Making my eigenvector matrix a 42 X number of modes required
a = zeros(42,40);
a(3:42,:) = phi_base(:,:);

%For loops generated to select the corresponding eigenvectors (magnitudes
%and displacements) out of the phi_base matrix and then multiplied them
%with the corresponding hermitian shape function second derivatives.
for i = 1:num_elements
   for k = 1:ndof-2
displeft(i,k) = a(2*i - 1,k);
rotleft(i,k) =  a(2*i,k);
dispright(i,k) = a(2*i + 1,k);
rotright(i,k) =  a(2*i + 2,k);

%"iprime" corresponds to the integral part of the strain energy equation
%already calculated in a polynomial form, with the corresponding hermitian second
derivative
%times its corresponding eigenvector product indicated.
iprime(i,k) = 1/3*(6/L^2*displeft(i,k)+2/L*rotleft(i,k)-
6/L^2*dispright(i,k)+4/L*rotright(i,k))^3/(12/L^3*displeft(i,k)+6/L^2*rotleft(i,k)-
12/L^3*dispright(i,k)+6/L^2*rotright(i,k))-1/3*(-6/L^2*displeft(i,k)-
4/L*rotleft(i,k)+6/L^2*dispright(i,k)-
2/L*rotright(i,k))^3/(12/L^3*displeft(i,k)+6/L^2*rotleft(i,k)-
12/L^3*dispright(i,k)+6/L^2*rotright(i,k));

strain(i,k) = iprime(i,k);
 end
end
% Assembling the strain energy equation:
UN = (nominal_EI/2)*strain
```

W. STRAINENERGYABC

```
%%%Strain Energy calculation of the base beam, with one ABC applied, not
sitffnes/mass perturbations.

clc
format long

%Given data
%E = 10e6;
%I1 = 0.016067021;
L = total_length/num_elements;

%Making my eigenvector matrix a 42 X number of modes required
aa = zeros(42,39);
Keep = [3:pinned - 1,pinned + 1:42];%This for 20 elements. The 42 (ndof) is to be
changed if # elements changes
aa(Keep,:) = phiaOSET(:,:);
%For loops generated to select the corresponding eigenvectors (magnitudes
%and displacements) out of the phi_base matrix and then multiplied them
%with the corresponding hermitian shape function second derivatives.
for i = 1:num_elements
   for k = 1:ndof-3
displeft(i,k) = aa(2*i - 1,k);
rotleft(i,k)  = aa(2*i,k);
dispright(i,k)= aa(2*i + 1,k);
rotright(i,k) = aa(2*i + 2,k);

%"iabcprime" corresponds to the integral part of the strain energy equation
%already calculated in a polynomial form, with the corresponding hermitian second
derivative
%times its corresponding eigenvector product indicated.

iabcprime(i,k) = 1/3*(6/L^2*displeft(i,k)+2/L*rotleft(i,k)-
6/L^2*dispright(i,k)+4/L*rotright(i,k))^3/(12/L^3*displeft(i,k)+6/L^2*rotleft(i,k)-
12/L^3*dispright(i,k)+6/L^2*rotright(i,k))-1/3*(-6/L^2*displeft(i,k)-
4/L*rotleft(i,k)+6/L^2*dispright(i,k)-
2/L*rotright(i,k))^3/(12/L^3*displeft(i,k)+6/L^2*rotleft(i,k)-
12/L^3*dispright(i,k)+6/L^2*rotright(i,k));

strain(i,k) = iabcprime(i,k);
 end
end
% Assembling the strain energy equation:
UNABC = (nominal_EI/2)*strain;
```

LIST OF REFERENCES

Allemang, R. J., Brown, D. L. A correlation coefficient for modal vector analysis. *Proceedings of 1st International Modal Analysis Conference*, 1982, 110-116.

Anton, H., Rorres, C. (2005). Elementary Linear Algebra. New York: John Wiley and Sons.

Avitable, Peter (2001, January) Experimental Modal Analysis, A Simple Non-Mathematical Presentation. *Sound and Vibration*, 1-11.

Brownjohn, J .M. W., Xia, Pin-Qi, Hao, Hong, & Xia, Yong. (2001) Civil structure condition assessment by FE model updating methodology and case studies. Finite *Elements in Analysis and Design 37* (211) 761-775.

Dascotte, E., Stroble, J., Hua, H. Sensitivity-Based Model Updating Using Multiple Types of Simultaneous State Variables. *13th International Modal Analysis Conference, Nashville, TN, 1995 pp.1035-1040.*

Ewins, D. J. (1984). *Modal Testing: Theory and Practice*. Research Studies Press LTD.

Ewins, D. J. (2000). *Modal Testing 2 .Theory, Practice and Application.* Research Studies Press LTD.

Flanigan, C. C. Test Analysis Correlation Using Design Sensitivity and Optimization-Does it work?. *Society of Automotive Engineers, Inc. 1988.*

Gordis, J. H. (1993) Spatial Frequency Domain Updating of Linear, Structural Dynamic Models.

Gordis, J. H. (1994, February 21). Structural Synthesis in the Frequency Domain: A General Formulation. *Shock and Vibration. 1*(5) 461-471.

Gordis, J. H. (1996, July). Omitted Coordinate Systems and Artificial Constraints in Spatially Incomplete Identification. *Modal Analysis: the International Journal of Analytical and Experimental Modal Analysis. 11*(1), 83-95.

Gordis, J. H. (1999) Artificial Boundary Conditions for Model Updating and Damage Detention. *Mechanical Systems and Signal Processing*, *13* (3), 437-448.

Hanson, D., Waters, T. P., Thompson, D. J., Randall, R. B., & Ford, R. A. J. (2006). The Role of Anti-Resonance Frequencies from Operating Modal Analysis in Finite Element Model Updating. *Mechanical Systems and Signal Processing 21*, 74-97.

Inman, D. J. (1989). *Vibration with Control Measurement and Stability.* Prentice Hall.

Jaishi, Bijaya, Ren, Wei-Xin (2005) Damage Detection by Finite Element Model Updating Using Modal Flexibility Residual. *Journal of Sound and Vibration 290*, 369-387.

Kenigsbuch, R. and Halevi, Y. (1998). Model Updating in Structural Dynamics: A generalized Reference Basis Approach. *Mechanical Systems and Signal Processing* 12, 75-90.

Kwon, Y. W., Bang, H. (2000). *The Finite Element Method Using MATLAB.* CRC Press Inc.

Luber, W., Sensburg, O. Identification of Errors and Updating in Analytical Models Using Test Data. *13th International Modal Analysis Conference, Nashville, TN, 1995 pp.407-413.*

Mottershead, J. E. (1993) Model Updating in Structural Dynamics: A Survey. *Journal of Sound and Vibration 167* (2), 347-375.

Panday, A. K., & Biswas, M. (1991). Damage Detection in Structures Using Changes in Flexibility. *Journal of Sound and Vibration 169* (1), 3-17.

Vanderplaats, G. *Numerical Optimization Techniques for Engineering Design.* McGraw-Hill, Inc. 1984.

Zhang, Q. W., Chang, C. C., & Chang, T. Y. P. (2000). Finite Element Model Updating for Structures with Parametric Constraints. *Earthquake Engineering and Structural Dynamics 29*, 927-944.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Mexican Navy General Staff
   Personnel and Training (B) Branch
   Mexico City, Mexico

4. Lt. Rafael A. Lagunes Arteaga
   Mexico City, Mexico